МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛ-ХОРАЗМИЙ

ФАКУЛЬТЕТ КОМПЬЮТЕРНОГО ИНЖИНИРИНГА

Кафедра «Компьютерных систем»

Ж.Х. Джуманов, Н.А. Сайфуллаева, С.Р. Ботиров, Р.Э. Яхшибоев

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения практических работ по предмету

ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

для студентов по направлении

«5330500» - Компьютерный инжиниринг («Компьютерный инжиниринг»)

ТУИТ имени Мухаммада аль-Хоразмий

Ташкент 2022

Авторы: Ж.Х. Джуманов, Н.А Сайфуллаева, С.Р. Ботиров, Р.Э. Яхшибоев «Компьютерные системы» / ТУИТ. 116 страниц. Ташкент 2021.

В методическом указании представлена последовательность практических работ по предмету «Интеллектуальный анализ данных», направленных на закрепление теоретических знаний полученных во время лекционных занятий и развитии практических навыков использования методов и средств интеллектуального анализа, при решении таких задач, как поиск, сбор, сортировка, классификация и оценка информации по изучаемой предметной области на основе прикладной программы Matlab. Каждая практическая работа выполняется студентом самостоятельно.

Методическое пособие предназначено для студентов бакалавриата 4 курса, обучающиеся по направлению 5330500 «Компьютерный инжиниринг» Ташкентского университета информационных технологий имени Мухаммада ал-Хоразмий.

Рассмотрен и одобрен на заседании Совета Ташкентского университета информационных технологий имени Мухаммада ал-Хоразмий

2022 год "_31_" _мая_, протокол № 10 (156)

Рецензенты:

Кафедра «Информационной безопасности»
ТУИТ имени Мухаммада ал-Хоразмий
К.т.н., доцент Ганиев А.А.

Практическая работа № 1
УСТАНОВКА И НАСТРОЙКА СРЕДЫ MATLAB.

Цель работы: Установка и запуска пакета MATLAB. Изучение особенностей MATLAB.

MATLAB (от англ. MATrix LABoratory) – язык программирования высокого уровня и система инженерных и научных вычислений, разработчиком которого является корпорация MathWorks Inc. (США). Система компьютерной математики MATLAB – сложный программный продукт. Его освоение целесообразно делать в два захода: вначале стоит изучить общие возможности системы и лишь затем приступить к основательному, нередко избранному знакомству с MATLAB.

Система MATLAB позволяет:

- выполнять математические вычисления;
- моделировать различные системы;
- анализировать данные, обрабатывать их и визуализировать;
- разрабатывать алгоритмы, приложения и пользовательский интерфейс.

При изучении математики используются в той или иной степени:

- *язык* MATLAB (Паскаль-и Си- подобный, объектно-ориентированный);
- *среда* MATLAB (командное окно, редактор, отладчик);
- *управляемая графика* (построение 2D, 3D графиков, создание анимации);
- *библиотека математических функций* (от самых простых ($\sin(x)$, $\cos(x)$ и т.п.) до более сложных, как нахождение обратной матрицы, собственных значений, экстремумов, производных и интегралов функций и множество других);
- *программный интерфейс* (набор встроенных процедур, функций и констант);

В среде MATLAB можно работать двумя способами:

1) непосредственным набором команд (в этом случае результаты вычисления присваиваются некоторым переменным);

2) с использованием программ, написанных на языке MATLAB, которая содержит все необходимые команды, обеспечивающие ввод данных, организацию вычислений и вывод результатов на экран (программный режим).

Выбор способа работы зависит от решаемой задачи. В обоих случаях используются практически все вычислительные возможности MATLAB. Преимущество второго способа состоит в возможности использования программ в вычислительных экспериментах

Для того, чтобы установить Matlab на компьютер клиента, вставьте инсталляционный DVD диск пакета Matlab в DVD-привод компьютера. Автоматически запустится мастер установки данного продукта. Если этого не произошло, запустите Setup.exe, расположенный в корневой директории инсталляционного диска Matlab.

Пример выполнения практической работы № 1

Шаг 1. После запуска некоторое время будут распаковываться нужные для установки файлы и в итоге откроется следующее окно:

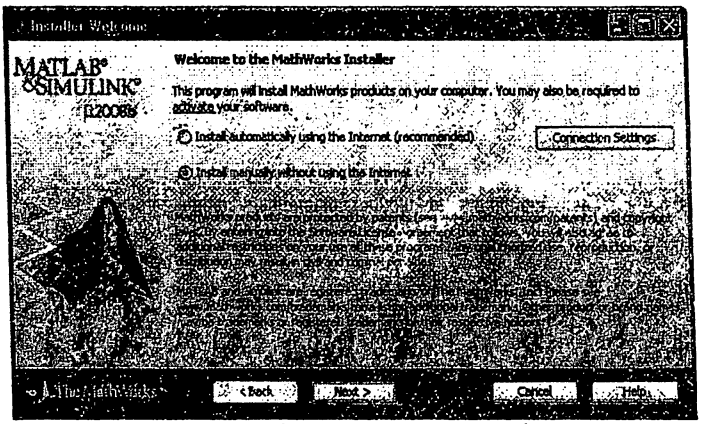


Рисунок 1.1 – Запуск установочного файла

Шаг 2. В этом окне выберите пункт «Install manually without using the Internet» (выборочная установка без использования Интернет), нажмите кнопку «Next>». Откроется окно лицензионного соглашения:

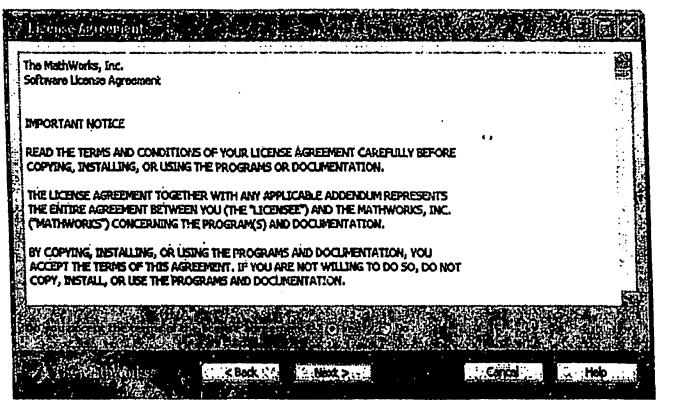


Рисунок 1.2 – Окно о принятии соглашения

Шаг 3. В этом окне согласитесь с условиями лицензии, выбрав пункт «Yes», нажмите кнопку «Next>». Откроется окно ввода инсталляционного ключа (файл с ключом получите вместе с инсталляционным диском, он называется fik.txt):

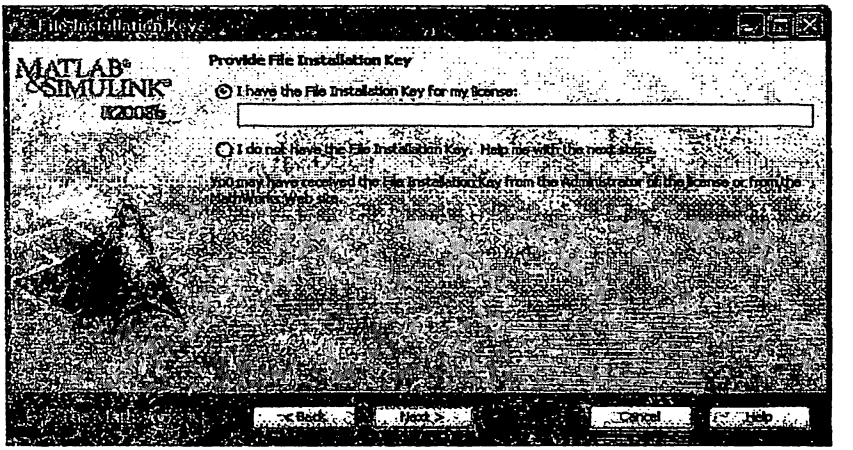


Рисунок 1.3 – Выбор установки с ключом.

Шаг 4. В этом окне выберите пункт «I have the File Installation Key for my license» (У меня есть файл с инсталляционным ключом для моей лицензии) и скопируйте этот ключ из полученного файла в поле, расположенное под этим пунктом. Нажмите кнопку «Next>». Откроется окно выбора установки: либо по умолчанию (Typical), либо настраиваемая (Custom):

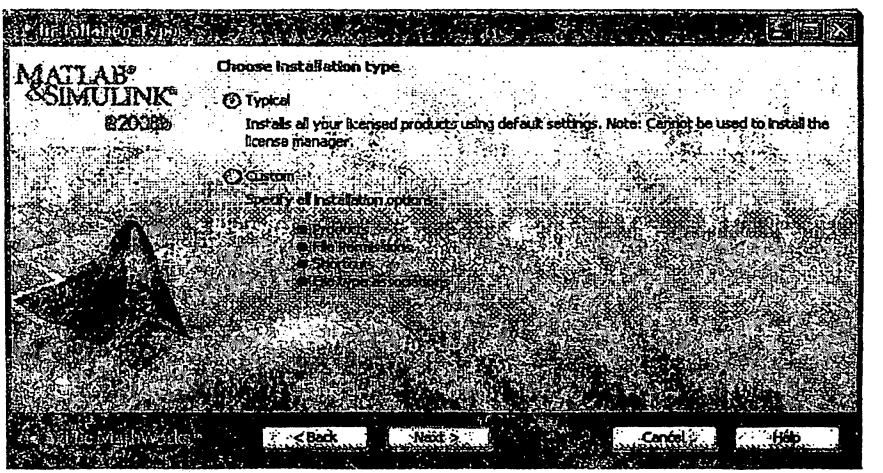


Рисунок 1.4 – Выбираем тип установки.

Шаг 5. В этом окне лучше выберите пункт Typical, нажмите кнопку «Next>». Откроется окно выбора папки для установки:

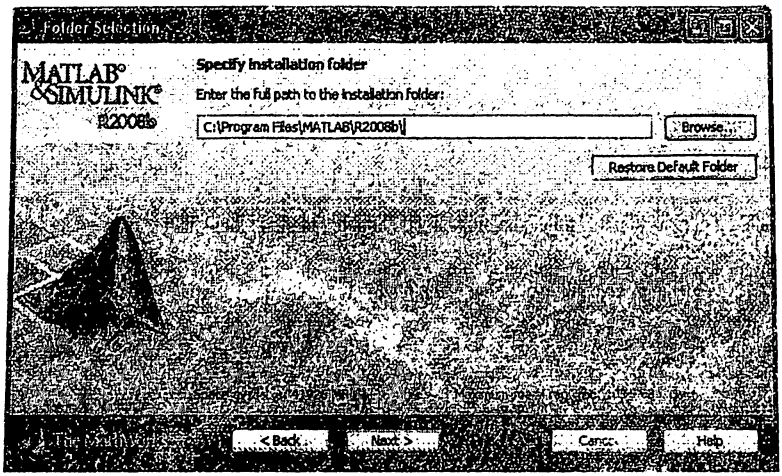


Рисунок 1.5 – Выбор папки установки

Шаг 6. В этом окне задайте полный путь для установки данного продукта, нажмите кнопку «Next>». Откроется окно выбора лицензионного файла (этот файл также будет выдан вместе инсталляционным диском, он называется – license.dat)

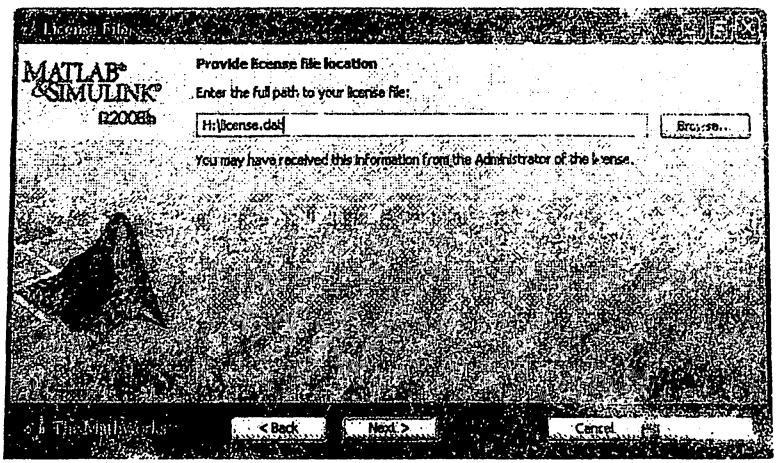


Рисунок 1.6 – Путь к файлу с лицензионным ключом

Шаг 7. В этом окне задайте полный путь до этого файла с лицензией, включая и название самого файла. Нажмите кнопку «Next>». Откроется окно процесса установки продукта:

Шаг 8. После завершения процесса установки откроется окно. Нажмите кнопку «Finish» для закрытия этого окна и завершения установки данного продукта.

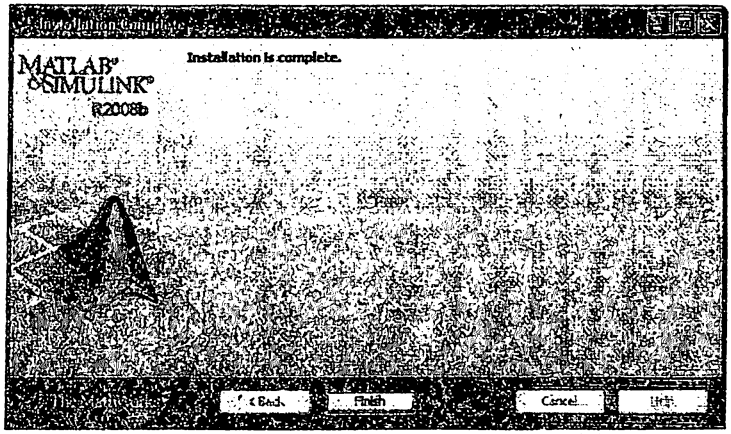


Рисунок 1.7 – Завершение установки Matlab

Для запуска MATLAB необходимо щелкнуть мышью по иконке программы. После чего на экране компьютера появится рабочее окно (рис. 1.9), в котором, как правило, расположены:

- панель инструментов, используемая для простоты и комфорта при работе с системой MATLAB (рис. 1.9);
- окно **Current Folder** (текущая папка), где показано содержимое текущей папки (рис. 1.11);
- окно **Command Window** (командное окно), в котором осуществляется ввод команд и вывод результата их выполнения;
- окно **Workspace** (рабочее пространство), которое содержит перечень текущих переменных и их описание;
- окно **Command History** (история команд), отображающее список ранее введенных команд.

В случае отсутствия одного из них в рабочем окне, их можно активизировать, например, через кнопку Layout вкладки HOME.

Назначение наиболее важных кнопок вкладки HOME (рис. 1.9):

- New Script (новый сценарий) открывает новое окно для работы в редакторе m-файлов;
- New (новый) позволяет создать новый объект (скрипт, функцию, её график и.т.п.);

- Open (открыть) открывает диалоговое окно для выбора m-файла, созданного ранее;
- Simulink открывает окно для работы в библиотеке Simulink;
- Layout (слои) и Preferences (свойства) – управление внешним видом программы;
- Help (помощь) – справка по работе с системой.

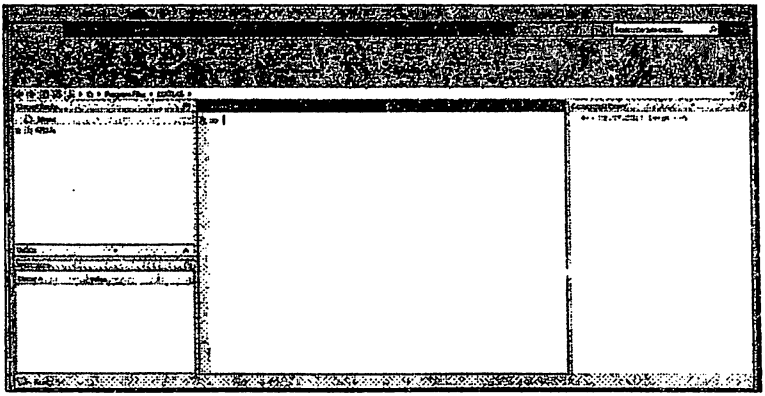


Рисунок 1.8 - Рабочее окно среды MATLAB при открытой вкладке HOME

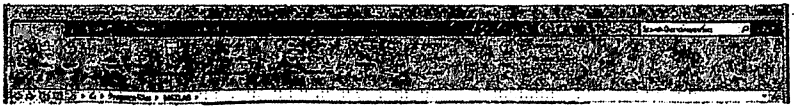
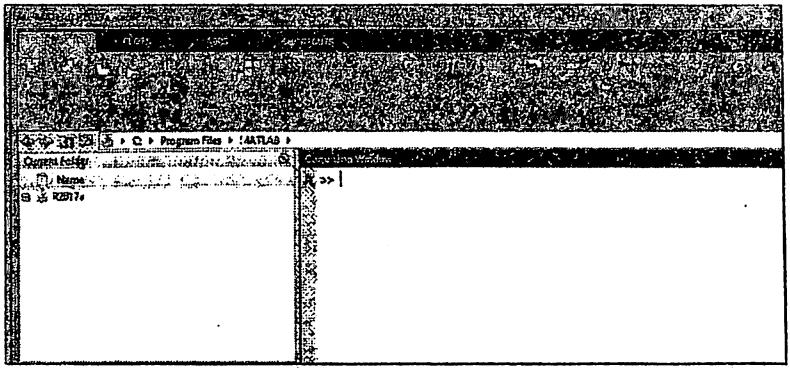
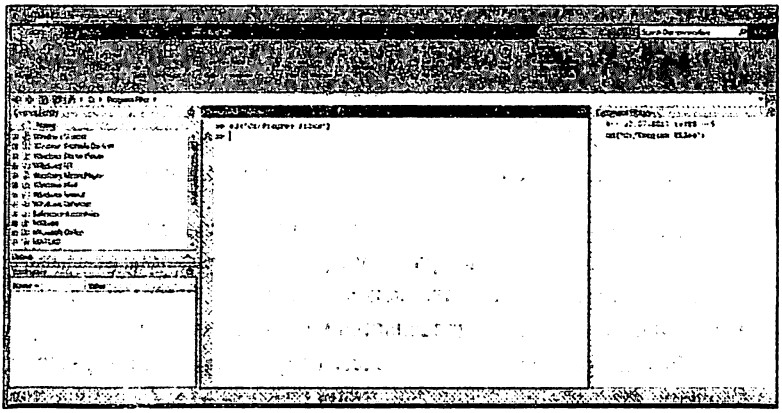


Рисунок 1.9 - Панель инструментов при открытой вкладке HOME



А



Б

Рисунок 1.10 - *А* – папка **Current Folder** и ее содержимое; *Б* – изменение текущего каталога: в строке ввода на панели инструментов, используя команду *cd <путь>* в **Command Window**

Задание для выполнения практической работы № 1

Задание 1. Необходимо установить и запустить пакет Matlab. Настройка Matlab.

Содержание и оформление отчета:

1. Наличие титульного листа, шрифт Times New Roman – 14, интервал 1,5;
2. Выполнение всех приложенных заданий;
3. Заключение по выполненной работе;
4. Ответы на контрольные вопросы.

Контрольные вопросы

1. Какие системных требований позволяет МАТЛАБ?
2. Сколько способов работ есть в среде МАТЛАБ?
3. Какие программный интерфейс имеется в среде МАТЛАБ?
4. Какие математические функции имеется в среде МАТЛАБ?

Практическая работа № 2
РАБОТА С ВЕКТОРНЫМИ И МАТРИЧНЫМИ ДАННЫМИ В MATLAB.

Цель работы: Изучение векторов и матриц, выполнение арифметических операций над векторами и матрицами в среде MATLAB.

MATLAB – среда, специально предназначенная для проведения сложных вычислений с векторами, матрицами и массивами. При этом она по умолчанию предполагает, что каждая заданная переменная – это вектор, матрица или массив. Все определяется конкретным значением переменной.

Матрица размера *1* x *N* называется *строкой* (или вектор-строкой), число элементов строки – ее *длиной*. Матрица размера *M* x *1* называется *столбцом* (или вектор-столбцом), число элементов столбца – его *высотой*.

Если задано X=1, то это значит, что X – это вектор с единственным элементом, имеющим значение 1, а точнее даже матрица с размером 1x1. Если надо задать вектор из трех элементов, то их значения следует перечислить в квадратных скобках, разделяя пробелами или запятыми:

```
>> V= [1 2 3]
     V = 1 2 3
```

задает вектор V, имеющий три элемента со значениями 1, 2 и 3 (его можно считать и матрицей размера 3_1). После ввода вектора система выводит его на экран дисплея. Заметим, для вектора столбца нужно разделять элементы знаками «;» (точка с запятой):

```
>> V=[1; 2; 3]
V =
        1
        2
        3
```

Задание матрицы требует указания нескольких строк и нескольких столбцов. Для разграничения строк используется знак ; (точка с запятой). Этот же знак в конце ввода предотвращает вывод матрицы или вектора (и вообще любой операции) на экран дисплея. Так, ввод

```
>> M=[1 2 3; 4 5 6; 7 8 9];
```

задает квадратную матрицу, которую можно вывести:

```
>> M
M =
     1     2     3
     4     5     6
     7     8     9
```

Возможен ввод элементов матриц и векторов в виде арифметических выражений, содержащих любые доступные в системе функции, например,:

```
>> V= [2+2/(3+4), exp(5), sqrt(10)];
>> V
     V = 2.2857        148.4132        3.1623
```

Для указания отдельного элемента вектора или матрицы используются выражения вида V(i) или M(i, j). Например, если задать

```
>> M(2, 2)
     ans = 5
```

то результат будет равен 5. Если нужно присвоить элементу M(i, j) новое значение x, следует использовать выражение

M(ij)=x

Например, если элементу M(2, 2) надо присвоить значение 10, следует записать

>> M(2, 2)=10

Вообще говоря, в тексте программ MATLAB лучше не использовать i и j как индексы, так как i и j – обозначение квадратного корня из –1. Но можно использовать I и J.

Выражение M(i) с одним индексом дает доступ к элементам матрицы, развернутым в один столбец. Такая матрица образуется из исходной, если подряд выписать ее столбцы. Следующий пример поясняет подобный доступ к элементам матрицы M:

```
>> M=[1 2 3; 4 5 6; 7 8 9]
M =
     1     2     3
     4     5     6
     7     8     9
>> M(2)
     ans = 4
>> M(8)
     ans = 6
>> M(9)
     ans = 9
>> M(5)=100;
>> M
M =
     1     2     3
     4   100     6
     7     8     9
```

Здесь уместно отметить, что размер векторов и матриц в данной книге учебного характера ограничен. Однако среда MATLAB способна работать с очень большими векторами и матрицами.

Наряду с операциями над отдельными элементами матриц и векторов среда позволяет производить операции умножения, деления и возведения в степень сразу над всеми элементами, то есть над массивами. Для этого перед знаком операции ставится точка. Например, оператор * означает умножение для векторов или матриц, а оператор .* – поэлементное умножение всех элементов массива.

Так, если M – матрица, то M.*2 даст матрицу, все элементы которой умножены на скаляр – число 2. Впрочем, для умножения матрицы на скаляр оба выражения– M*2 и M.*2 – оказываются эквивалентными.

Имеется также ряд особых функций для задания векторов и матриц. Например, функция magic(n) задает магическую матрицу размера *n_n*, у которой сумма всех столбцов, всех строк и даже диагоналей равна одному и тому же числу:

```
>> M=magic(4)
M =
    16    2    3    13
     5   11   10     8
     9    7    6    12
     4   14   15     1
>> sum(M)
ans = 34  34   34   34
>> sum(M')
     ans = 34  34   34   34
>> sum(diag(M))
     ans = 34
>> M(1,2)+M(2,2)+M(3,2)+M(4,2)
     ans = 34
```

Уже сама по себе возможность создания такой матрицы с помощью простой функции magic заинтересует любителей математики. Но векторных и матричных функций в среде множество, и мы их детально рассмотрим в дальнейшем.

Описанный способ задания матриц позволяет выполнить операцию *конкатенации* – объединения малых матриц в большую матрицу. Например, создадим вначале магическую матрицу размера 3_3:

```
>> A=magic(3)
A =
```

```
     8     1     6
     3     5     7
     4     9     2
```

Теперь можно построить матрицу, содержащую четыре матрицы:

```
>> B=[A A+16;A+32 A+16]
B =
     8     1     6    24    17    22
     3     5     7    19    21    23
     4     9     2    20    25    18
    40    33    38    24    17    22
    35    37    39    19    21    23
    36    41    34    20    25    18
```

Полученная матрица имеет уже размер 6_6. Вычислим сумму ее столбцов:

```
>> sum(B)
     ans = 126 126 126 126 126 126
```

Любопытно, что она одинакова для всех столбцов. А для вычисления суммы строк используем команду

```
>> sum(B.')
     ans = 78 78 78 174 174 174
```

Здесь запись B.' означает транспонирование матрицы B, то есть замену строк столбцами. На этот раз сумма оказалась разной. Это отвергает изначально возникшее предположение, что матрица «B» тоже является магической. Для истинно магической матрицы суммы столбцов и строк должны быть одинаковыми:

```
>> D=magic(6)
D =
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
>> sum(D) = sum(D.')
     ans = 111  111      111  111  111  111
```

Более того, для магической матрицы одинаковой является и сумма элементов по основным диагоналям (главной диагонали и главной антидиагонали).

Для формирования матриц и выполнения ряда матричных операций возникает необходимость удаления отдельных столбцов и строк матрицы. Для

этого используются пустые квадратные скобки – [ ]. Проделаем это с матрицей M:

```
>> M=[1 2 3; 4 5 6; 7 8 9]
M =
     1     2     3
     4     5     6
     7     8     9
```

Удалим второй столбец, используя оператор: (двоеточие):

```
>> M(:,2)=[ ]
M =
     1     3
     4     6
     7     9
```

А теперь, используя оператор: (двоеточие), удалим вторую строку:

```
>> M(2,:)=[ ]
M =
     1     3
     7     9
```

ОПЕРАЦИИ НАД МАТРИЦАМИ

Для работы с матрицами в MATLAB существуют специальные команды и функции. Основные операции с матрицами приведены в табл. 3.

Операции с матрицами

Таблица 2.1

Команда	Описание
A+B	Сложение матриц A и B одинаковой размерности
A–B	Вычитание матриц A и B одинаковой размерности
*A*B*	Матричное произведение массивов ($A_{n\times m}$ $B_{m\times k}$=$C_{n\times k}$)
*A.*B*	Поэлементное умножение матриц A и B одинаковой размерности
A./B	Поэлементное деление матриц A и B одинаковой размерности
A.^k	Поэлементное возведение массива в степень k
A'	Транспонирование матрицы A
A^–1 *inv(A)*	Вычисление обратной матрицы (A^{-1})

Параметры матриц

Таблица 2.2

det(A)	Вычисление определителя матрицы (|A|)
size(A)	Определение размерности матрицы A
trace(A)	След матрицы A (сумма элементов на главной диагонали)
sum(A)	Сумма элементов в каждом столбце матрицы A
prod(A)	Произведение элементов в каждом столбце матрицы A
diag(A)	Вектор-столбец элементов главной диагонали матрицы A
sort(A)	Сортировка каждого столбца матрицы A
max(A)	Вычисление максимума в каждом столбце матрицы A
min(A)	Вычисление минимума в каждом столбце матрицы A
mean(A)	Вычисление среднего значения в каждом столбце матрицы A
rot90(A)	Поворот матрицы A влево на 90
fliplr(A)	Отразить матрицу A слева направо
flipud(A)	Отразить матрицу A сверху вниз
A(n,:) = []	Удаление из матрицы A строки с номером n
A(:,n) = []	Удаление из матрицы A столбца с номером n

Пример выполнения практической работы № 2

ОПЕРАЦИИ НАД ВЕКТОРАМИ

```
%Создание вектора с помощью команды rand
>> V1= floor(rand(1,3)*10)
V1 =
     6     7     7
%Ввод вектора вручную
>> V2= [3, 6, 1]
V2 =
     3     6     1
%Операции над векторами
>> A=V1+V2

A =
     9    13     8
```

```
>> B=V1-V2

B =
     3     1     6

>> C=V1.*V2

C =

    18    42     7

>> C=V1./V2
C =
    2.0000    1.1667    7.0000

>> C=floor(V1./V2)

C =

     2     1     7
```

MATLAB имеет много функций, которые создают различные виды матриц. Например, можно создать симметричную матрицу с записями на основе треугольника, можно задать матрицу с числами, которые будут сгенерированы наугад самой программой, можно задать элементы матрицы вручную.

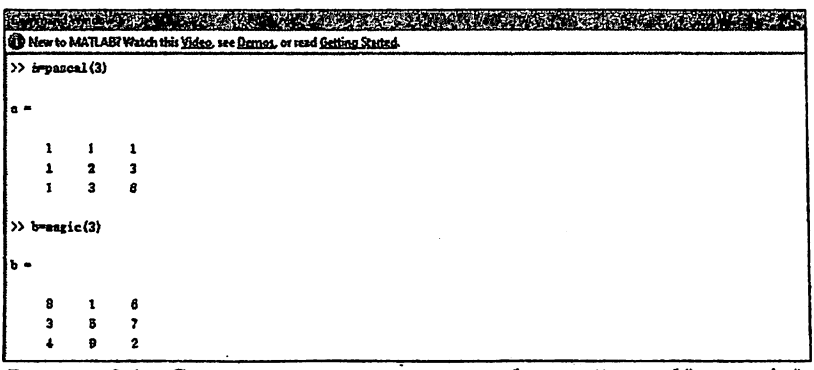


Рисунок 2.1 – Создание матриц при помощи функций pascal() и magic()

Ввод матрицы вручную.

```
>> A= [3 2 1; 1 4 -3; 7 2 0]

A =

     3     2     1
     1     4    -3
     7     2     0

>> B = [4 1 2; -1  1 0; 6 2 1]

B =

     4     1     2
    -1     1     0
     6     2     1
```

Рисунок 2.2 – Создание матриц A и B вручную

Далее выполним операции над матрицами:

```
>> Z = A+B

Z =

     7     3     3
     0     5    -3
    13     4     1
```

```
>> H = A-B

H =

    -1     1    -1
     2     3    -3
     1     0    -1
```

```
>> X = A*B

X =

    16     7     7
   -18    -1    -1
    26     9    14
```

```
>> Y = B*A

Y =

    27    16     1
    -2     2    -4
    27    22     0
```

Рисунок 2.3 – Сложение, вычитание и умножение матриц

Задание для выполнения практической работы № 2

Задание 1. Необходимо вручную (без помощи команды **magic()**) построить матрицу «A» размером 4x4 (таблица 2.3) и задать случайную матрицу «B» размером 4x4, используя команды Matlab (pascal, magic, rand).

Задание 2. Необходимо выполнить все арифметические действия над построенным матрицами.

Задание 3. Задать произвольные вектора (одинокой размерности, вектор строка или вектор столбец) и выполнить все арифметические действия с векторами.

Варианты для выполнения практической работы № 2

Таблица 2.3

1.	$\begin{bmatrix} 2 & 3 & -4 & 1 \\ 1 & -3 & -1 & 2 \\ 3 & 3 & -7 & 0 \\ 1 & -12 & 1 & 5 \end{bmatrix}$	8.	$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 4 & -1 & 3 & -1 \\ 2 & -7 & -1 & -3 \\ 6 & 5 & 7 & 1 \end{bmatrix}$
2.	$\begin{bmatrix} 1 & -3 & 2 & 1 \\ 4 & 1 & -2 & 3 \\ 2 & 7 & 6 & 1 \\ 5 & -2 & 0 & 4 \end{bmatrix}$	9.	$\begin{bmatrix} 1 & 3 & 5 & 1 \\ 3 & 5 & 3 & 5 \\ 1 & -1 & -4 & 3 \\ 2 & 4 & 4 & 3 \end{bmatrix}$
3.	$\begin{bmatrix} 3 & -1 & 1 & -2 \\ 2 & 1 & -2 & 1 \\ 1 & -2 & 3 & -3 \\ 0 & -5 & 8 & -7 \end{bmatrix}$	10.	$\begin{bmatrix} 1 & -4 & 1 & -2 \\ 2 & 3 & -1 & 1 \\ 4 & -5 & 1 & -3 \\ 1 & -15 & 4 & -7 \end{bmatrix}$
4.	$\begin{bmatrix} 3 & 4 & -2 & 3 \\ 2 & 7 & -5 & -1 \\ 1 & -3 & 3 & 4 \\ 4 & 1 & 1 & 7 \end{bmatrix}$	11.	$\begin{bmatrix} 3 & -1 & 2 & -5 \\ 2 & -3 & 3 & -2 \\ 5 & -4 & 5 & -7 \\ 1 & 2 & -1 & -2 \end{bmatrix}$
5.	$\begin{bmatrix} 1 & 3 & 2 & -1 \\ 2 & -1 & 3 & 2 \\ 3 & -5 & 4 & 5 \\ 1 & 10 & 6 & -6 \end{bmatrix}$	12.	$\begin{bmatrix} 2 & -3 & 4 & 2 \\ 1 & -1 & 3 & -1 \\ 4 & -7 & 11 & 3 \\ 3 & -5 & 5 & 5 \end{bmatrix}$
6.	$\begin{bmatrix} 5 & -1 & 2 & -3 \\ 1 & -2 & 1 & -1 \\ 3 & 3 & 0 & -1 \\ 2 & 5 & -1 & 0 \end{bmatrix}$	13.	$\begin{bmatrix} 2 & 5 & 3 & -4 \\ 2 & -1 & -1 & 3 \\ 6 & -7 & -5 & -1 \\ 0 & 5 & -1 & -5 \end{bmatrix}$
7.	$\begin{bmatrix} 3 & 4 & 2 & -5 \\ 4 & 1 & -4 & 1 \\ 2 & 7 & 8 & -11 \\ -1 & 3 & 6 & -6 \end{bmatrix}$	14.	$\begin{bmatrix} 1 & -3 & -5 & 7 \\ 2 & -1 & -3 & 4 \\ 1 & 2 & 2 & -3 \\ 1 & -8 & -12 & -7 \end{bmatrix}$

Содержание и оформление отчета:

1. Наличие титульного листа, шрифт Times New Roman – 14, интервал 1,5;
2. Выполнение всех приложенных заданий;
3. Заключение по выполненной работе;
4. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Какие математические операторы используется в среде МАТЛАБ?
2. Какие операции можно выполнять в среде МАТЛАБ?
3. Какая функция используется для магической матрицы в среде МАТЛАБ??
4. Какие операции выполняется над матрицами в среде МАТЛАБ?

Практическая работа № 3
ВИЗУАЛИЗАЦИЯ И ПОСТРОЕНИЕ ГРАФИКОВ ДАННЫХ В MATLAB.

Цель работы: Научиться строить двухмерные, трехмерные и пользовательские графики в среде MATLAB

В состав MATLAB входит мощная графическая подсистема, которая поддерживает как средства визуализации двумерной и трехмерной графики на экран терминала, так и средства презентационной графики. Следует выделить несколько уровней работы с графическими объектами. В первую очередь это команды и функции, ориентированные на конечного пользователя и предназначенные для построения графиков в прямоугольных и полярных координатах, гистограмм и столбцовых диаграмм, трехмерных поверхностей и линий уровня, анимации.

Графические команды высокого уровня автоматически контролируют масштаб, выбор цветов, не требуя манипуляций со свойствами графических объектов. Соответствующий низкоуровневый интерфейс обеспечивается дескрипторной графикой, когда каждому графическому объекту ставится в соответствие графическая поддержка (дескриптор), на который можно ссылаться при обращении к этому объекту. Используя дескрипторную графику, можно создавать меню, кнопки вызова, текстовые панели и другие объекты графического интерфейса.

Элементарные графические функции среды MATLAB позволяют построить на экране и вывести на печатающее устройство следующие типы графиков: линейный, логарифмический, полулогарифмический, полярный.

Для каждого графика можно задать заголовок, нанести обозначение осей и масштабную сетку.

Двумерные графики

- PLOT - график в линейном масштабе
- LOGLOG - график в логарифмическом масштабе
- SEMILOGX, SEMILOGY - график в полулогарифмическом масштабе
- POLAR - график в полярных координатах

Трехмерные графики

В среде MATLAB предусмотрено несколько команд и функций для построения трехмерных графиков. Значения элементов числового массива рассматриваются как z-координаты точек над плоскостью, определяемой координатами x и y. Возможно несколько способов соединения этих точек. Первый из них - это соединение точек в сечении (функция plot3), второй - построение сетчатых поверхностей (функции mesh и surf). Поверхность,

построенная с помощью функции mesh, - это сетчатая поверхность, ячейки которой имеют цвет фона, а их границы могут иметь цвет, который определяется свойством EdgeColor графического объекта surface.

Поверхность, построенная с помощью функции surf, - это сетчатая поверхность, у которой может быть задан цвет не только границы, но и ячейки; последнее управляется свойством FaceColor графического объекта surface. Уровень изложения данной книги не требует от читателя знания объектно-ориентированного программирования. Ее объем не позволяет в полной мере описать графическую подсистему, которая построена на таком подходе. Заинтересованному читателю рекомендуем обратиться к документации по среде MATLAB, и в первую очередь к только что вышедшей из печати книге Using MATLAB Graphics (Natick, 1996).

- MESHGRID - формирование двумерных массивов X и Y
- PLOT3 - построение линий и точек в трехмерном пространстве
- MESH, MESHC, MESHZ - трехмерная сетчатая поверхность
- SURF, SURFC - затененная сетчатая поверхность
- SURFL - затененная поверхность с подсветкой
- AXIS - масштабирование осей и вывод на экран
- GRID - нанесение сетки
- HOLD - управление режимом сохранения текущего графического окна
- SUBPLOT - разбиение графического окна
- ZOOM - управление масштабом графика
- COLORMAP - палитра цветов
- CAXIS - установление соответствия между палитрой цветов и масштабированием осей
- SHADING - затенение поверхностей
- CONTOURC - формирование массива описания линий уровня
- CONTOUR - изображение линий уровня для трехмерной поверхности
- CONTOUR3 - изображение трехмерных линий уровня

Специальная графика

Раздел специальной графики включает графические команды и функции для построения столбцовых диаграмм, гистограмм, средств отображения векторов и комплексных элементов, вывода дискретных последовательностей данных, а также движущихся траекторий как для двумерной, так и для трехмерной графики. Этот раздел получил свое дальнейшее развитие в версии системы MATLAB 5.0, где специальные графические средства улучшены и существенно расширены.

- BAR - столбцовые диаграммы
- ERRORBAR - график с указанием интервала погрешности
- HIST - построение гистограммы
- STEM - дискретные графики
- STAIRS - ступенчатый график
- ROSE - гистограмма в полярных координатах
- COMPASS, FEATHER - графики векторов
- QUIVER - поле градиентов функции
- COMET - движение точки по траектории
- FILL - закраска многоугольника
- COMET3 - движение точки по пространственной траектории
- SLICE - сечения функции от трех переменных
- WATERFALL - трехмерная поверхность
- FILL3 - закраска многоугольника в трехмерном пространстве
- VIEWMTX - вычисление матрицы управления углом просмотра
- VIEW - управление положением точки просмотра

ДВУМЕРНЫЕ ГРАФИКИ

PLOT - график в линейном масштабе

Синтаксис:

```
plot(y)
plot(x, y)
plot(x, y, s)
plot(x1, y1, s1, x2, y2, s2, ...)
```

Описание:

Команда plot(y) строит график элементов одномерного массива y в зависимости от номера элемента; если элементы массива y комплексные, то строится график plot(real(y), imag(y)). Если Y - двумерный действительный массив, то строятся графики для столбцов; в случае комплексных элементов их мнимые части игнорируются.

Команда plot(x, y) соответствует построению обычной функции, когда одномерный массив x соответствует значениям аргумента, а одномерный массив y - значениям функции. Когда один из массивов X или Y либо оба двумерные, реализуются следующие построения:

- если массив Y двумерный, а массив x одномерный, то строятся графики для столбцов массива Y в зависимости от элементов вектора x;
- если двумерным является массив X, а массив y одномерный, то строятся графики столбцов массива X в зависимости от элементов вектора y;

- если оба массива X и Y двумерные, то строятся зависимости столбцов массива Y от столбцов массива X.

Команда plot(x, y, s) позволяет выделить график функции, указав способ отображения линии, способ отображения точек, цвет линий и точек с помощью строковой переменной s, которая может включать до трех символов из следующей таблицы:

Способы отображения линий двумерных графиков

Таблица 3.1.

Тип линии		Тип точки		Цвет	
Непрерывная	-	Точка	.	Желтый	y
Штриховая	--	Плюс	+	Фиолетовый	m
Двойной пунктир	:	Звездочка	*	Голубой	c
Штрих-пунктирная	-.	Кружок	o	Красный	r
		Крестик	x	Зеленый	g
				Синий	b
				Белый	w
				Черный	k

Если цвет линии не указан, он выбирается по умолчанию из шести первых цветов, с желтого до синего, повторяясь циклически.

Команда plot(x1, y1, s1, x2, y2, s2, ...) позволяет объединить на одном графике несколько функций y1(x1), y2(x2), ..., определив для каждой из них свой способ отображения.

Обращение к командам plot вида plot(x, y, s1, x, y, s2) позволяет для графика y(x) определить дополнительные свойства, для указания которых применения одной строковой переменной s1 недостаточно, например при задании разных цветов для линии и для точек на ней.

Пример выполнения практической работы № 3

Построим график функции y = sin(x) на отрезке [-p p] с шагом p /500:

```
>>x = -pi:pi/500:pi;
>>y = sin(x);
>>plot(x, y) % рис. 3.1.
```

График на рис. а отображает значения одномерного массива y, состоящего из 1001 элемента, как функцию от номера элемента; график на рис. б отображает значения того же массива как функцию элементов массива x.

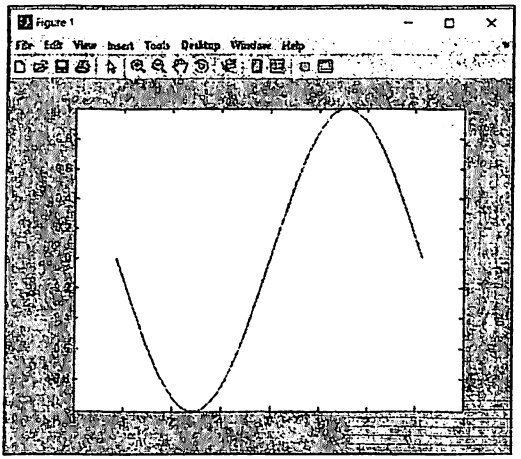


Рисунок 3.1 - График функции y

Рассмотрим различные способы применения функции plot(x, y) на примере графиков двух функций y1 = sin(x) и y2 = xsin(x):

```
x1 = -pi:pi/500:pi;
y1 = sin(x);
y2 = x.*sin(x);
plot(x',[y' y2']) % рис. 3.2.
```

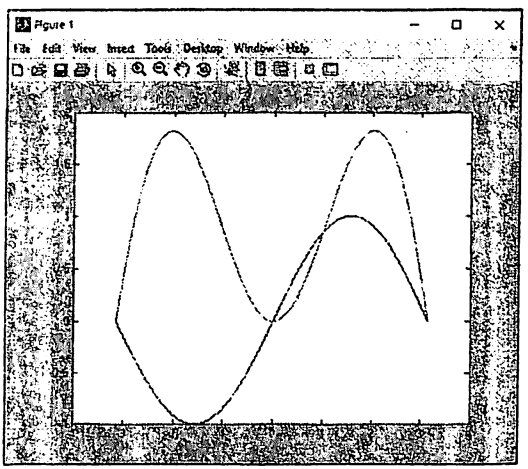


Рисунок 3.2 - графики функций y и y2 в одном окне

```
x2 = x/2;
y2 = x2.*sin(x2);
plot([x' x2'], [y' y2']) % рис. 3.3
```

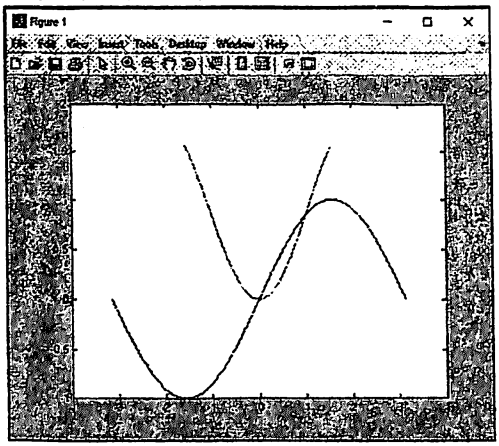


Рисунок 3.3 - Графики функций y и y2 с разными x

POLAR - график в полярных координатах

Синтаксис:

polar(phi, rho) polar(phi, rho, s)

Описание:

Команды polar(...) реализуют построение графиков в полярных координатах, задаваемых углом phi и радиусом pho.

Примеры:

Построим график функции rho = sin(2 * phi) * cos(2 * phi) в полярных координатах

```
>> phi = 0:0.01:2 * pi;
>> polar(phi, sin(2 * phi). * cos(2 * phi))
```

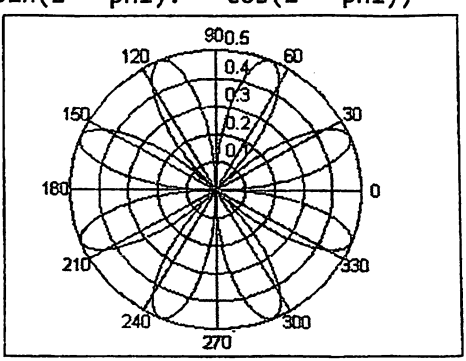


Рисунок 3.4 - График в полярных координатах

ТРЕХМЕРНЫЕ ГРАФИКИ

PLOT3 - построение линий и точек в трехмерном пространстве

Синтаксис:

```
plot3(x, y, z)
plot3(X, Y, Z)
plot3(x, y, z, s)
plot3(x1, y1, z1, s1, x2, y2, z2, s2, ...)
```

Описание:

Команды plot3(...) являются трехмерными аналогами функции plot(...).

Команда plot3(x, y, z), где x, y, z - одномерные массивы одинакового размера, строит точки с координатами x(i), y(i), z(i) и соединяет их прямыми линиями.

Команда plot3(x, y, z, s) позволяет выделить график функции z(x, y), указав способ отображения линии, способ отображения точек, цвет линий и точек с помощью строковой переменной s, которая может включать до трех символов из следующей таблицы.

Примеры:

Построим график функции z = x * exp(-x2 - y2) в трехмерном пространстве.

```
>> [ X, Y ] = meshgrid([ -2 : 0.1 : 2 ]);
>> Z = X.* exp(- X .^ 2 - Y .^ 2);
>> plot3(X, Y, Z)
```

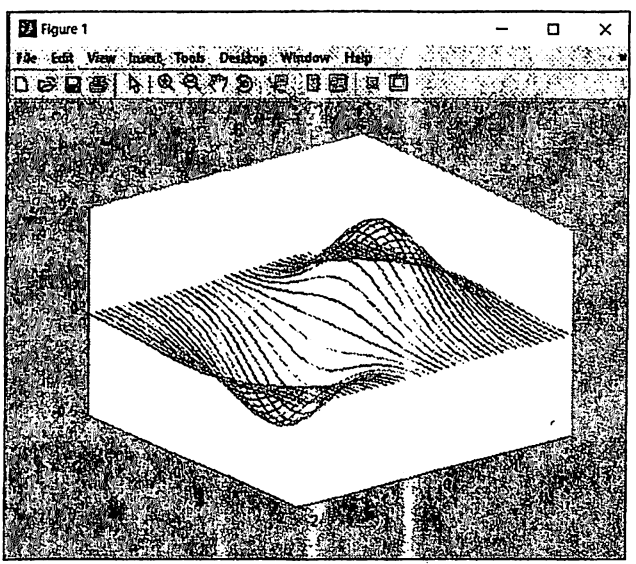


Рисунок 3.5 – Трехмерный график, построенный с помощью функции plot3()

MESH, MESHC, MESHZ - трехмерная сетчатая поверхность

Синтаксис:

```
mesh(X, Y, Z, C)   meshc(X, Y, Z, C)   meshz(X, Y, Z, C)
mesh(x, y, Z, C )  meshc(x, y, Z, C )  meshz(x, y, Z, C )
mesh(Z, C)         meshc(Z, C)         meshz(Z, C)
mesh(X, Y, Z)      meshc(X, Y, Z)      meshz(X, Y, Z)
mesh(x, y, Z)      meshc (x, y, Z)     meshz(x, y, Z)
mesh(Z)            meshc(Z)            meshz(Z)
```

Описание:

Команда mesh(X, Y, Z, C) выводит на экран сетчатую поверхность для значений массива Z, определенных на множестве значений массивов X и Y. Цвета узлов поверхности задаются массивом C. Цвета ребер определяются свойством EdgeColor объекта surface. Можно задать одинаковый цвет для всех ребер, определив его в виде вектора [r g b] интенсивности трех цветов - красного, зеленого, синего. Если определить спецификацию none, то ребра не будут прорисовываться. Если определить спецификацию flat, то цвет ребер ячейки определяется цветом того узла, который был первым при обходе этой ячейки. Поскольку одни и те же ребра обходятся несколько раз, то цвета будут замещаться. Если определить спецификацию interp, то будет реализована линейная интерполяция цвета между вершинами ребра.

Пример выполнения практической работы № 3

Построим график функции z = x * exp(-x2 - y2) в трехмерном пространстве.

```
>> [ X, Y ] = meshgrid([ -2 : 0.1 : 2 ]);
>> Z = X .* exp(- X .^ 2 - Y .^ 2);
mesh(X, Y, Z)
```

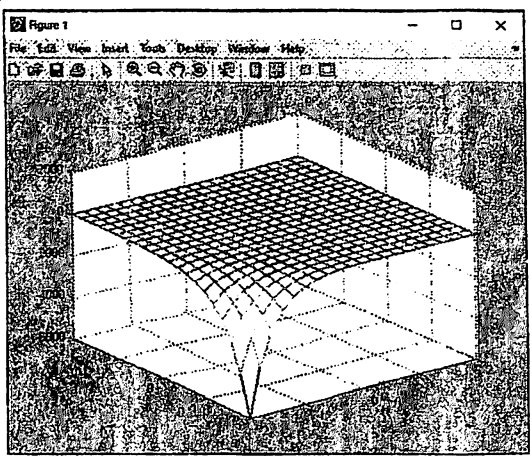


Рисунок 3.6 – Трехмерный график, построенный с помощью функции mesh()

SURF, SURFC - затененная сетчатая поверхность

Синтаксис:

```
surf(X, Y, Z, C)        surfc(X, Y, Z, C)
surf(x, y, Z, C )       surfc(x, y, Z, C )
surf(Z, C)              surfc(Z, C)
surf(X, Y, Z)           surfc(X, Y, Z)
surf(x, y, Z)           surfc (x, y, Z)
surf(Z)                 surfc(Z)
```

Описание:

Команда surf(X, Y, Z, C) выводит на экран сетчатую поверхность для значений массива Z, определенных на множестве значений массивов X и Y. Цвет ячейки определяется массивом C. Цвет ребер - черный, определяется свойством EdgeColor, специфицированным как [0 0 0]. Можно задать одинаковый цвет для всех ребер, определив его в виде вектора [r g b] интенсивности трех цветов - красного, зеленого, синего. Если определить спецификацию none, то ребра не будут прорисовываться.

Построим график функции z = x * exp(-x2 - y2) в трехмерном пространстве.

```
>> [ X, Y ] = meshgrid([ -2 : 0.1 : 2 ]);
>> Z = X .* exp(- X .^ 2 - Y .^ 2);
>> surfc(X, Y, Z)
```

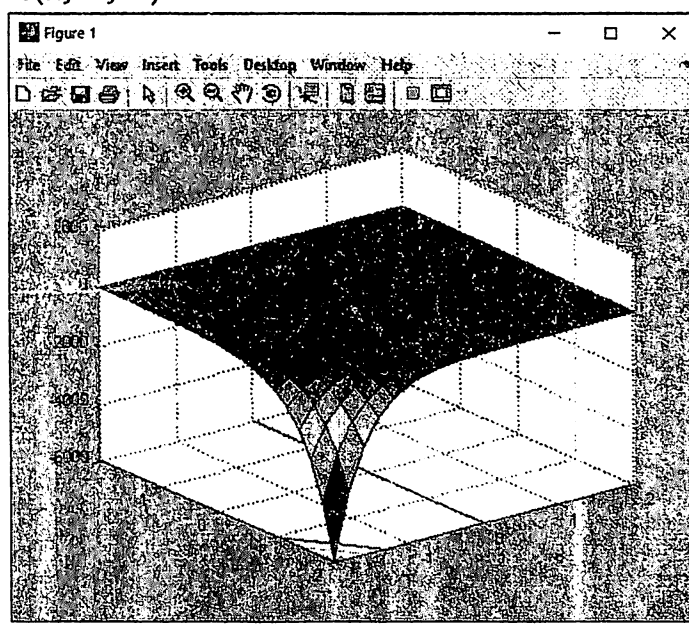


Рисунок 3.7 – Трехмерный график, построенный с помощью функции surfc()

СПЕЦИАЛЬНЫЕ ГРАФИКИ

BAR - столбцовые диаграммы

Синтаксис:

```
bar(y)bar(x, y)
[xb, yb] = bar(...)
bar(y, '<тип линии>')
bar(x, y, '<тип линии>')
```

Описание:

Команда bar(y) выводит график элементов одномерного массива y в виде столбцовой диаграммы.

Команда bar(x, y) выводит график элементов массива y в виде столбцов в позициях, определяемых массивом x, элементы которого должны быть упорядочены в порядке возрастания.

Если X и Y - двумерные массивы одинаковых размеров, то каждая диаграмма определяется соответствующей парой столбцов и они надстраиваются одна над другой.

Команды bar(y, '<тип линии>'), bar(x, y, '<тип линии>') позволяют задать тип линий, используемых для построения столбцовых диаграмм, по аналогии с командой plot.

Функция [xb, yb] = bar(...) не выводит графика, а формирует такие массивы xb и yb, которые позволяют построить столбцовую диаграмму с помощью команды plot(xb, yb).

Примеры:

Построить график функции $y = e^{-x^2}$ в виде столбцовой диаграммы.

```
>> x = -2.9 : 0.2 : 2.9;
>> bar(x, x+1)
```

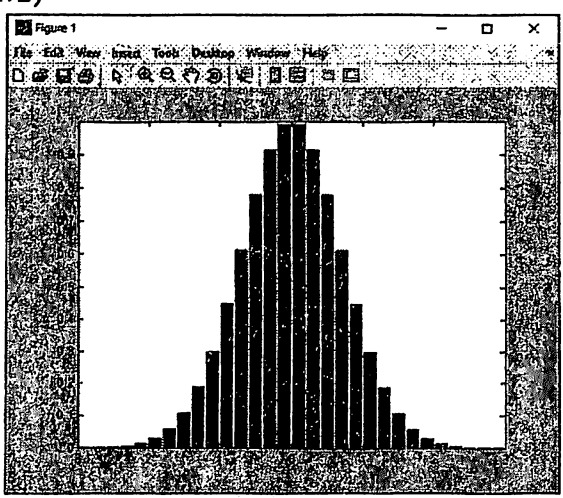


Рисунок 3.8 – График, построенный с помощью функции bar()

Задание для выполнения практической работы № 3

Задание 1. По заданному варианту, выбрать функцию, необходимо вывести значения x и y. Построить 2D график функции, вывести на экран Matlab.

Задание 2. Далее задать вторую произвольную функцию и вывести график её функции вместе с графиком функции по варианту, на экране Matlab.

Задание 3. По заданному варианту, выбрать функцию, необходимо вывести значения x, y, z в командую строку Matlab. Построить 3D график функции, вывести на экран Matlab, используя следующие команды форматирования графиков: **plot3(); grid(); mesh(); surfc(); surfl().**

Варианты для постройки двумерных графиков

Таблица 3.2.

№	$f_1(x)$	$f_2(x)$	Interval
1.	$y=\frac{e^{-x}}{e^{-x}+1}-x$	$y=\frac{4x}{4+x^2}$	$x\in(0;5),\Delta x=0.01$
2.	$y=\frac{x^2-1}{\ln(x^2-1)}+x$	$y=\frac{x^2+1}{x^2-1}$	$x\in(3;7),\Delta x=0.01$
3.	$y=x^3-12x+7$	$y=\sin x+\sin x-1$	$x\in(0;3),\Delta x=0.01$
4.	$y=3x^4-16x^3+2$	$y=\frac{x^3}{(x^3+1)}$	$x\in[-3;1],\Delta x=0.01$
5.	$y=x-\sin x$	$y=\frac{(4x^3+5)}{x}$	$x\in(-\pi;\pi),\Delta x=\frac{\pi}{32}$
6.	$y=\frac{x^2}{4}+1,-1\le x\le 1$	$y=\frac{x}{(2x^2+1)}$	$x\in(-5;5),\Delta x=0.02$
7.	$y=2x^2+1,-1\le x\le 1$	$y=xe^{-x^2}$	$x\in(-3;3),\Delta x=0.01$
8.	$y=(x^2-1)e^{3x+1}$	$y=(x+1)^2,0\le x\le 2$	$x\in(-3;4),\Delta x=0.02$
9.	$y=\frac{\ln x}{\sqrt{x}}$	$y=x^5-0.6x^3+1$	$x\in[2;5],\Delta x=0.02$
10.	$y=(x^2+1)^{\cos x},0\le x\le 1$	$y=81x-x^4$	$x\in[0;2\pi],\Delta x=\frac{\pi}{16}$
11.	$y=(2+x^2)e^{-x}$	$y=x^3,0\le x\le 2$	$x\in[0;5],\Delta x=0.02$
12.	$y=\sin x+(x-2),0\le x\le\pi$	$y=\frac{x^3}{(x-1)}$	$x\in[0;2\pi],\Delta x=\frac{\pi}{32}$
13.	$y=\frac{1}{2}(x+1)^2,-1\le x\le 0$	$y=2^x e^{-x}$	$x\in[0;4\pi],\Delta=\frac{\pi}{32}$
14.	$y=tgx+\frac{x}{2},0\le x\le\frac{\pi}{4}$	$y=x-\sin x$	$x\in[-\pi;\pi],\Delta x=\frac{\pi}{32}$
15.	$y=\frac{\sqrt{3}}{2}x+\cos x$	$y=(x^2+2)e^{-(x+1)}$	$x\in[0;2\pi],\Delta x=\frac{\pi}{16}$

Варианты для построения трехмерных графиков

Таблица 3.3.

1	$z = x^2 + xy + y^2$	[-5:0.2:5]
2	$z = 3x^2 - xy + x + y$	[-5:0.2:5]
3	$z = x^2 + 3xy - 6y$	[-5:0.2:5]
4	$z = x^2 - y^2 + 6x + 3y$	[-5:0.2:5]
5	$z = x^2 + 2xy + 3y^2$	[-5:0.2:5]
6	$z = x^2 + y^2 + 2x + y - 1$	[-5:0.2:5]
7	$z = 3x^2 + 2y^2 - xy$	[-5:0.2:5]
8	$z = 2xy + 3y^2 - 5x$	[-5:0.2:5]
9	$z = 2xy + 3y^2 - 5x$	[-5:0.2:5]
10	$z = xy + 2y^2 - 2x$	[-5:0.2:5]
11	$z = x^2 + y^2 - 9xy + 27$	[-5:0.2:5]
12	$z = x^2 + 2y^2 + 1$	[-5:0.2:5]
13	$z = 3 - 2x^2 - xy - y^2$	[-5:0.2:5]
14	$z = x^2 + 3y^2 + x - y$	[-5:0.2:5]
15	$z = 2xy + 3y^2 + 5x$	[-5:0.2:5]
16	$z = 3x^2 + 2y^2 - xy$	[-5:0.2:5]
17	$z = 2xy + 3y^2 - 5x$	[-5:0.2:5]
18	$z = xy - 2y^2 - 2x$	[-5:0.2:5]
19	$z = x^2 + y^2 + 9xy + 27$	[-5:0.2:5]
20	$z = x^2 - 2y^2 + 1$	[-5:0.2:5]
21	$z = 3 - 2x^2 - xy - y^2$	[-5:0.2:5]
22	$z = x^2 + 3y^2 + 3x - y$	[-5:0.2:5]
23	$z = x^2 + 2xy + 3y^2 - 5$	[-5:0.2:5]
24	$z = x^2 + 2xy + y^2 - 5$	[-5:0.2:5]
25	$z = x^2 + xy - y^2$	[-5:0.2:5]
26	$z = 3x^2 + xy + x + y$	[-5:0.2:5]
27	$z = x^2 + 3xy - 6y$	[-5:0.2:5]
28	$z = x^2 - y^2 - 6x + 3y$	[-5:0.2:5]
29	$z = x^2 - 2xy + 3y^2$	[-5:0.2:5]
30	$z = x^2 + y^2 - 2x + y - 1$	[-5:0.2:5]

Содержание и оформление отчета:

1. Наличие титульного листа, шрифт Times New Roman – 14, интервал 1,5;
2. Выполнение всех приложенных заданий;
3. Заключение по выполненной работе;
4. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Какие графические функции имеется в среде МАТЛАБ?
2. Какие специальные графики имеется в среде МАТЛАБ?
3. Какие команды используется в двумерном графике в среде МАТЛАБ?
4. Какие команды используется в трехмерном графике в среде МАТЛАБ?

Практическая работа № 4
ЗАГРУЗКА ДАННЫХ. СПОСОБЫ ИМПОРТА И ЭКСПОРТА ДАННЫХ В MATLAB (PANDAS, IMPORTDATA)

Цель работы: Изучение способов импорта и экспорта данных в среде MATLAB

При обработке данных возникает необходимость хранения как исходных данных, так и результатов вычислений. Для этого обычно используются файлы. *Файлы* – это довольно распространенные объекты среды MATLAB. Последнее, в частности, относится к файлам данных. О некоторых типах файлов уже говорилось в предшествующих уроках.

Открытие и закрытие файла

Перед использованием любого файла он должен быть *открыт*, а по окончании использования – *закрыт*. Много файлов может быть открыто и доступно для чтения одновременно. Рассмотрим команды открытия и закрытия файлов. Команда open имя, где имя должно содержать массив символов или символьную переменную, открывает файлы в зависимости от анализа параметра имя и расширения в имени имя:

- переменная – открывает массив, названный по имени, в редакторе массивов (**Array Editor**);
- .mat – открывает файл, сохраняет переменные в структуре в рабочей области;
- .fig – открывает его в редакторе дескрипторной графики Property Editor;
- .m – открывает m_файл в редакторе_отладчике;
- .mdl – открывает модель в Simulink;
- .p – открывает, если он есть, m_файл с тем же именем;
- .html – открывает HTML_документ в браузере помощи.

Если файлы с расширением существуют в пути MATLAB, то открывается тот файл, который возвращается командой which имя, если нет – то файл из файловой системы. Если файл не имеет расширения имени, то он открывается той программой, формат файлов которой был бы обнаружен функцией which('имя файла'). По умолчанию для всех файлов с окончаниями, отличными от вышеперечисленных, вызывается openother. Open вызывает функции openxxx, где xxx – расширение файла. Исключение – переменные рабочей области, для которых вызывается openvar, и рисунки, для работы с которыми вызывается openim. Создавая m_файлы с именем openxxx, пользователи могут изменять обработку файлов и добавлять новые

расширения в список. Закрывать файлы, открытые при помощи open, нужно из редакторов, вызываемых openxxx.

- [FILENAME, PATHNAME] = uigetfile(FILTERSPEC, Title). Открывает диалог с именем Title и фильтром FILTERSPEC (например, массивом ячеек, содержащим расширения файлов) и возвращает файл, выбранный пользователем, и путь к нему. Возвращает FILENAME=0, если файл не существует или если пользователь нажал на Cancel. [FILENAME, PATHNAME] = uigetfile (FILTERSPEC, Title, X, Y) размещает окно диалога в точке *X*, *Y* (координаты в пикселях). Пример: [filename, pathname] = uigetfile('*.m;*.fig;*.mat;*.mdl', 'All MATLAB Files (*.m, *.fig, *.mat, *.mdl)'; ...
- [FILENAME, PATHNAME] = uiputfile(FILTERSPEC, TITLE) сохраняет файл в диалоге, управляемом пользователем. Параметры аналогичны таковым в функции uigetfile.

Функция saveas сохраняет рисунок или модель Simulink в желаемом формате на носителе информации или на устройстве, разрешенном print. Функция saveas (H,'FILENAME') сохраняет данные в соответствии с командой дескрипторной графики H в файле FILENAME. Формат файла определяется расширением имени FILENAME.

Функция saveas (H,'FILENAME','FORMAT') выполняет то же, но с параметром FORMAT (формат задается тем же способом, что и расширение имени файла, и может от него отличаться). FORMAT имеет приоритет перед расширением имени файла. Параметры:

- 'fig' – сохранить рисунок (график) в двоичном fig-файле;
- 'm' или 'mfig' – сохранить рисунок в двоичном fig-файле и создать m-файл для его загрузки;
- 'mmat' – сохранить рисунок в m-файле как последовательность команд создания рисунка. Может не поддерживать новейших графических функций.

Примеры:

saveas(gcf, 'output', 'fig')

saveas(gcf, 'output', 'bmp')

Команда или функция delete удаляет файл или объект графики. delete имя файла удаляет файл текущей папки. Может быть использована *.

Функция close(H) закрывает только графические окна. Для закрытия файлов необходимо использовать команду fclose.

Для записи файлов на диск служит команда save, используемая в довольно очевидных формах:

```
save          save filename          save filename var1 var2 ...
save ... option                      save('filename', ...)
```

Соответственно, для считывания файлов с диска служит команда load:

```
load                    load filename        load filename X Y Z
load filename -ascii    load filename -mat   S = load(...)
```

В этих командах имя файла указывается по правилам, принятым в операционных системах класса MS_DOS. Эти команды обычно дублируются кнопками панелей инструментов и браузером файлов.

Двоичными, или *бинарными*, называют файлы, данные которых представляют собой машинные коды. Основные операции с такими кодами перечислены ниже.

- fopen(filename, permission) открывает файл с именем filename и параметром, определенным в permission, и возвращает идентификатор fid со значением: 0 -- чтение с клавиатуры (permission установлено в 'r'); 1 – вывод на дисплей (permission установлено в 'a'); 2 – вывод сообщения об ошибке (permission установлен в 'a'); –1 – неудача в открытии файла с выводом сообщения message о типе ошибки. Идентификатор fid часто используется в качестве аргумента другими функциями и программами ввода_вывода. Имя файла filename может содержать путь к файлу.

  Если открываемый для чтения файл не найден в текущем каталоге, то функция fopen осуществляет поиск файла по пути, указанном в MATLAB. Параметр permission может принимать одно из следующих основных значений (другие см. в справочной системе):

  — 'r' – открытие файла для чтения (по умолчанию);
  — 'r+' – открытие файла для чтения и записи;
  — 'w' – удаление содержимого существующего файла или создание нового и открытие его для записи;
  — 'a' – создание и открытие нового файла или открытие существующего для записи с добавлением в конец файла.

  Добавление к этой строке 'b' (подразумевается по умолчанию) предписывает системе открыть файл в двоичном режиме.

  Добавление же вместо b к этой строке 't', например 'rt', в операционных системах, которые имеют различие между текстовыми и двоичными файлами, предписывает системе открыть файл в текстовом режиме. Например, во всех версиях MATLAB для Windows/MS_DOS и VMS нельзя открыть текстовый файл без параметра 'rt'. При вводе файлов с использованием fopen в текстовом режиме удаляются все символы «возврат каретки» перед символом новой строки.

- [fid,message] = fopen(filename,permission,format) открывает файл, как описано выше, возвращая идентификатор файла и сообщение. Кроме того, значение параметра format позволяет точно определить числовой формат. Возможны 8 форматов, описание которых можно найти в справочной системе. В частности, строка format может иметь значения 'native' (формат компьютера, на котором установлена система), 'vax', 'cray' (компьютеры VAX и Cray) и т. д.

  Определенные вызовы функций fread или fwrite могут отменить числовой формат, заданный при вызове функции fopen.
- fids = fopen('all') возвращает вектор_строку, содержащую идентификаторы всех открытых файлов, не включая стандартные потоки 0, 1 и 2. Число элементов вектора равно числу открытых пользователем файлов.
- [filename,permission,format] = fopen(fid) возвращает полное имя файла, строку permission и строку format. При использовании недопустимых значений fid возвращаются пустые строки для всех выходных аргументов.
- Команда fclose закрывает файл. Она имеет следующие варианты.
- status = fclose(fid) закрывает файл, если он открыт. Возвращает статус файла status, равный 0, если закрытие завершилось успешно, и –1 в противном случае. Аргумент fid – это идентификатор, связанный с открытым файлом (см. функцию fopen для более подробного описания).
- status = fclose('all') закрывает все открытые файлы. Возвращает 0 в случае успешного завершения и –1 в противном случае.

Пример открытия и закрытия файла:

```
>> fid=fopen('c:\ex','a+')
fid = 4
>> fclose(4)
ans = 0
```

Функция **importdata** позволяет загружать различные файлы данных разных форматов. Он имеет следующие пять форм ––

1. **A = importdata (имя файла)** Загружает данные в массив A из файла, обозначенного *именем файла* .
2. **A = importdata ('- pastespecial')** Загружает данные из системного буфера обмена, а не из файла.
3. **A = importdata (___, delimiterIn)** Интерпретирует *delimiterIn* как *разделитель* столбцов в файле ASCII, имени файла или данных буфера обмена. Вы можете использовать *delimiterIn* с любым из входных аргументов в приведенных выше синтаксисах.

4. **A = importdata (___, delimiterIn, headerlinesIn)** Загружает данные из файла ASCII, имени файла или буфера обмена, считывая числовые данные, начиная со строки *headerlinesIn + 1* .
5. **[A, delimiterOut, headerlinesOut] = importdata (___)** Возвращает обнаруженный символ разделителя для входного файла ASCII в *delimiterOut* и обнаруженное количество строк заголовка в *headerlinesOut* , используя любой из входных аргументов в предыдущих синтаксисах.

Пример выполнения практической работы № 4

Примеры записи и чтения матрицы:

```
>> fid = fopen('c:\prim','a+')
fid = 3
>> A=magic(7)
A =
     30   39   48    1    10   19   28
     38   47    7    9    18   27   29
     46    6    8   17    26   35   37
      5   14   16   25    34   36   45
     13   15   24   33    42   44    4
     21   23   32   41    43    3   12
     22   31   40   49     2   11   20
>> count = fwrite(3,A)
count = 49
>> status = fclose(3)
status = 0
>> fid = fopen('c:\prim','r')
fid = 3
>> [B,count] = fread(3,[7,7])
B =
     30   39   48    1    10   19   28
     38   47    7    9    18   27   29
     46    6    8   17    26   35   37
      5   14   16   25    34   36   45
     13   15   24   33    42   44    4
     21   23   32   41    43    3   12
     22   31   40   49     2   11   20
count = 49
```

Пример экспорта изображения из Matlab в корневую папку Matlab:

```
>> surf(peaks)
```

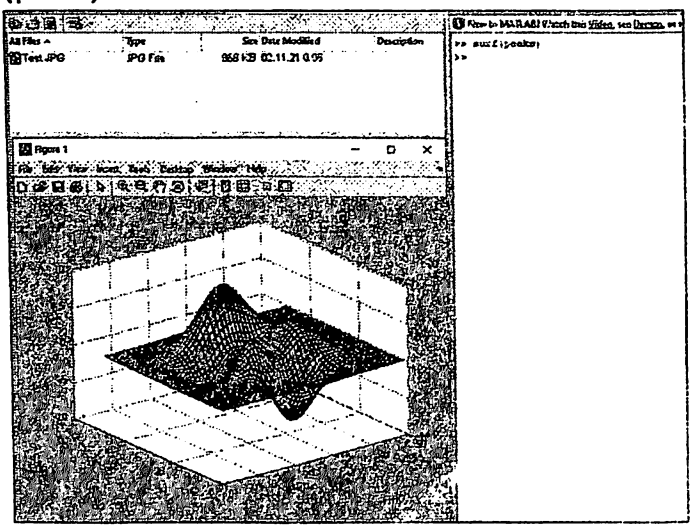


Рисунок 4.1 - Создаем трехмерный график

Используем команду gcf, чтобы получить указатель текущей фигуры:

```
>> fig = gcf;
>> saveas(gcf, '3D grafik', 'jpg')
```

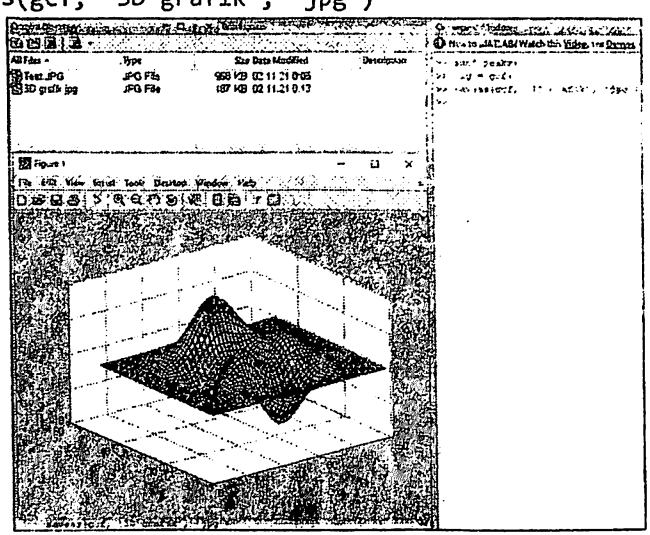


Рисунок 4.2 – Используем команду «gfc», и экспортируем изображение.

Пример импорта изображения в Matlab:

Загрузим и отобразим файл изображения. Создайте файл сценария и введите в нем следующий код —

```
>> filename = 'Test.jpg';
>> TestImage = importdata(filename);
>> image(TestImage);
```

Когда вы запускаете файл, Matlab отображает файл изображения. Однако вы должны сохранить его в текущем каталоге.

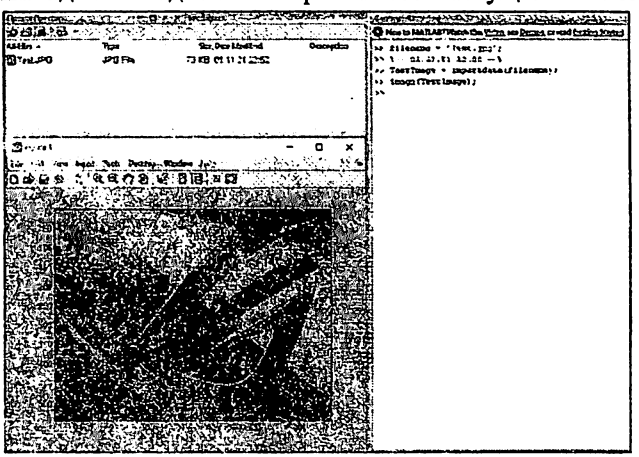


Рисунок 4.3 - Импорт изображения в Matlab. Пример 1.

Пример 2. Указываем путь к изображению которое необходимо импортировать:

```
>> TestImage=importData('C:\Test.jpg',1);
>> image(TestImage)
```

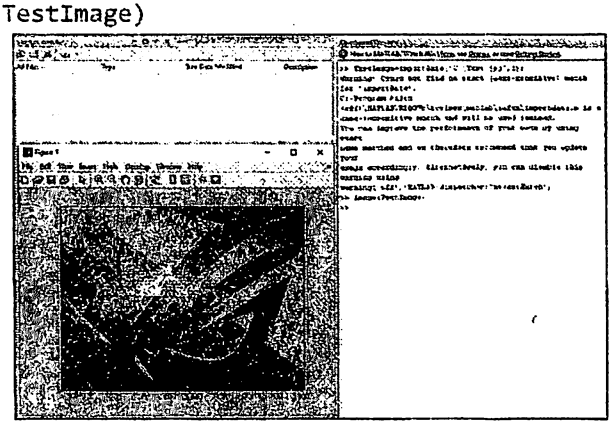


Рисунок 4.4 - Импорт изображения в Matlab. Пример 2.

Пример импорта текстовой информации

В этом примере мы импортируем текстовый файл и указываем разделитель и заголовок столбца. Давайте создадим ASCII-файл, разделенный пробелами, с заголовками столбцов с именем *TestFile.txt* .

Создаем файл сценария и введем в нем следующий код —

```
>> filename = 'TestFile.txt';
>> delimiterIn = ' ';
>> headerlinesIn = 1;
>> TestImport =
importdata(filename,delimiterIn,headerlinesIn);
>> for k = [1:7]
disp(TestImport.colheaders{1, k})
disp(TestImport.data(:, k))
disp(' ')
end
```

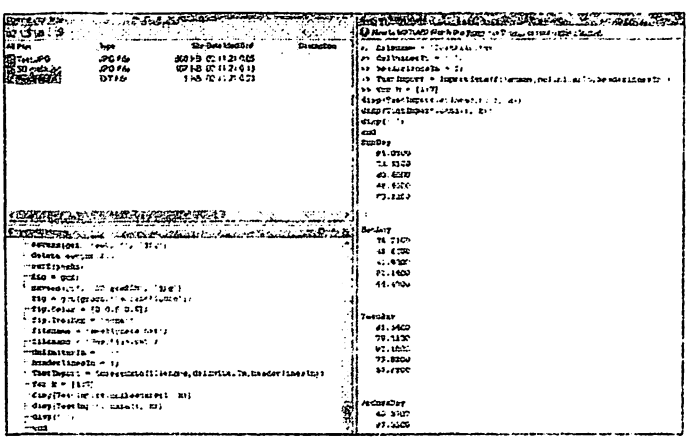


Рисунок 4.6 – Результат импорта данных из файла в командную строку.

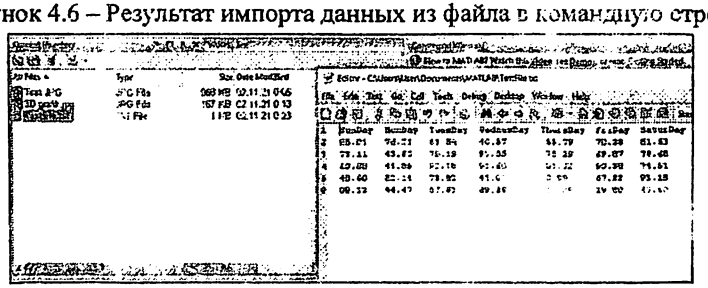


Рисунок 4.5 - Созданный текстовый файл TestFile.txt, загруженный в корневую папку Matlab

Импорт данных из текстового файла

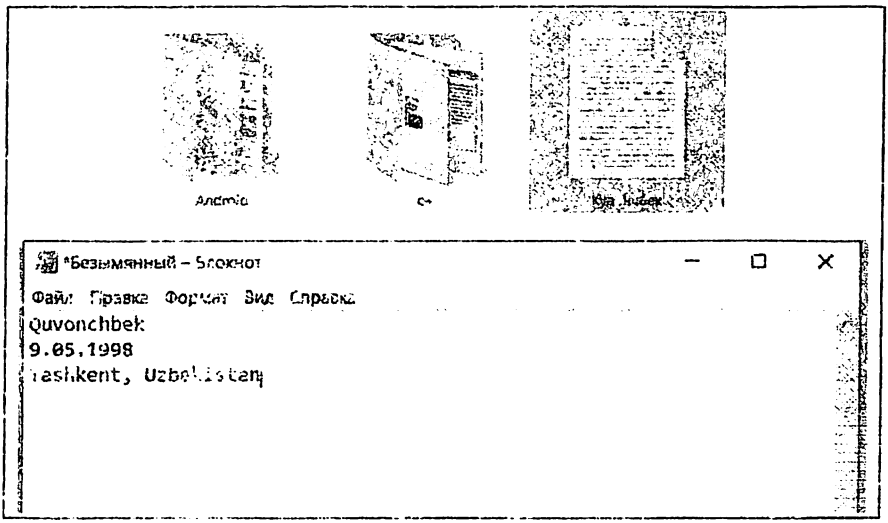


Рисунок 4.7 – Данные в текстовом файле

```
>> delimiterIn=' ';
>> headerlinesIn=2; %строк заголовка
>> A = importdata ('C:\Desktop\Кувончбек.txt');
>> A
A =
    3x1 cell array
        {'Quvonchbek '     }
        {'9.05.1998'           }
        {'Tashkent, Uzbekistan'}
>>data = A.data
data =
9    5    1998
>> textdata = A.textdata
textdata =
    1x1 cell array
    {'Quvonchbek '     }
```

Экспорт данных в текстовый файл

```
>> x=[1:10]
x =
```

```
    1    2    3    4    5    6    7    8    9    10
>> y=5.*x;
>> z=7.*x;
>> dlmwrite ('C:\Desktop\newtxt.txt', [x,y,z], 'delimiter', '\t');
```

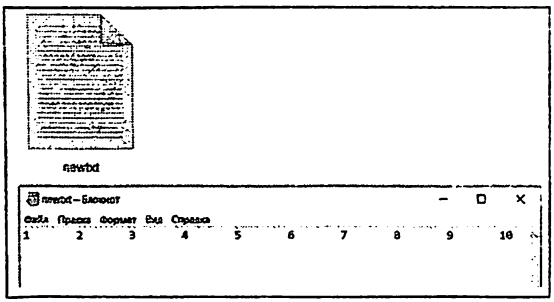


Рисунок 4.8 – Данные экспортированные в текстовый файл

Задание для выполнения практической работы № 4

Задание 1. Построить график используя произвольную функцию (2D график, 3D график, специальный график), построенный график экспортировать в корневую папку Matlab.

Задание 2. Создать произвольную матрицу, экспортировать её в файл с указанием пути к папке с файлом, после чего импортировать записанную матрицу в Matlab.

Задание 3. Импортировать изображение в Matlab, которое расположено в корневой папке Matlab, и импортировать изображение с указанием пути к файлу, который находится вне корневой папки.

Задание 4. Прочитать данные, заранее записанные в текстовый документ и сохраненный в корневой папке Matlab, полученные данные вывести в командной строке Matlab.

Содержание и оформление отчета:

1. Наличие титульного листа, шрифт Times New Roman – 14, интервал 1,5;
2. Выполнение всех приложенных заданий;
3. Заключение по выполненной работе;
4. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Какие команды используется при открытии файла в среде МАТЛАБ?
2. Какие команды используется при закрытии файла в среде МАТЛАБ?
3. Что такое операция PANDAS в среде МАТЛАБ?
4. Что такое операция IMPORT DATA в среде МАТЛАБ?

Практическая работа № 5
АНАЛИЗ И ФИЛЬТРАЦИЯ ДАННЫХ В MATLAB

Цель работы: Изучение методов анализа данных и фильтрация данных разного типа в среде MATLAB

MATLAB может использоваться для моделирования работы цифровых фильтров. Для обеспечения *дискретной одномерной фильтрации* используется функция filter в следующих формах записи.

- *filter(B,A,X)* фильтрует одномерный массив данных X, используя дискретный фильтр, описываемый следующим конечноразностным уравнением:
  a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb) - a(2)*y(n-1) – ... – a(na+1)*y(n-na). Если a(1) не равно 1, то коэффициенты уравнения нормализуются относительно a(1). Когда X – матрица, функция filter оперирует столбцами X. Возможна фильтрация многомерного (размерности *N*) массива *[Y,Zf]=filter(B,A,X,Zi)* выполняет фильтрацию с учетом ненулевого начального состояния фильтра Zi; возвращает, помимо выходного сигнала Y, конечное состояние фильтра Zf.
- *filter(B,A,X,[ ],dim)* или *filter(B,A,X,Zi,dim)* работают в направлении размерности dim.

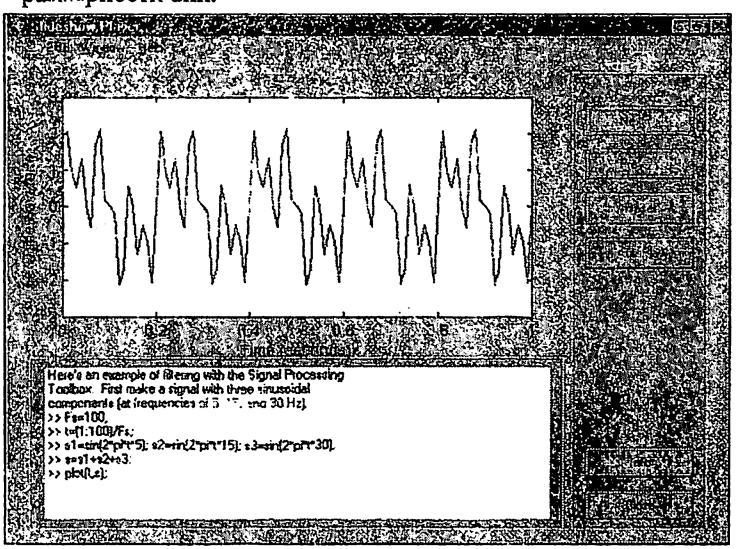


Рисунок 5.1 - Формирование сигнала и построение его графика

Рассмотрим типовой пример фильтрации гармонического сигнала на фоне других сигналов – файл с именем **filtdem.m** из пакета расширения Signal Processing Toolbox. На рис. 5.1 представлен кадр примера, на котором показано формирование входной совокупности сигналов в виде трех сигналов с частотами 5, 15 и 30 Гц. Показаны временная зависимость сигнала и под ней фрагмент программы, включающий команды, которые надо выполнить для этого этапа примера.

Следующий кадр (рис. 5.2) иллюстрирует конструирование фильтра с доста_точно плоской вершиной амплитудночастотной характеристики (АЧХ) и поло_ сой частот, обеспечивающего выделение сигнала с частотой 15 Гц и подавление сигналов с частотами 5 и 30 Гц. Для формирования полосы пропускания фильтра используется функция ellip, а для построения АЧХ – функция freqz (обе – из пакета Signal Processing Toolbox). Это позволяет построить график АЧХ созданного фильтра.

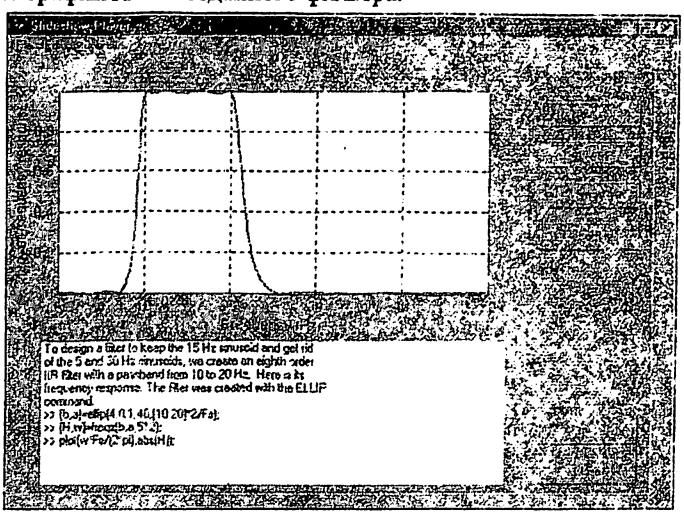


Рисунок 5.2 - Конструирование фильтра с заданной полосой частот и построение графика его АЧХ

Следующий кадр примера (рис. 5.3) иллюстрирует эффективность выделения сигнала заданной частоты (15 Гц) с помощью операции фильтрации – функции filter, описанной выше. Можно заметить два обстоятельства – полученный стационарный сигнал практически синусоидален, что свидетельствует о высокой степени фильтрации побочных сигналов. Однако нарастание сигнала во времени идет достаточно медленно и занимает несколько периодов частоты полезного сигнала. Характер

нарастания сигнала во времени определяется переходной характеристикой фильтра.

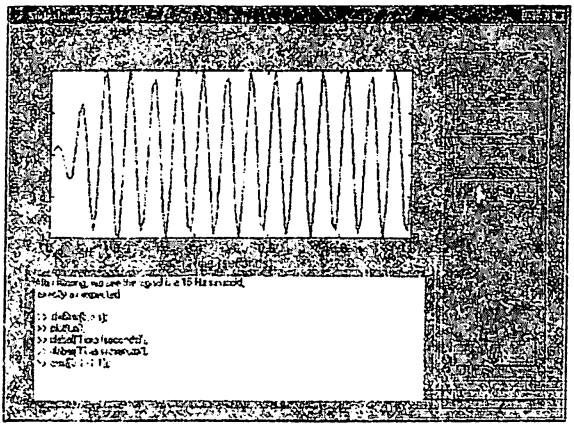


Рисунок 5.3 - Фильтрация и ее результат в виде временной зависимости сигнала на выходе фильтра

Заключительный кадр (рис. 5.4) показывает спектр исходного сигнала и спектр сигнала на выходе фильтра (он показан линиями другого цвета, что, к сожалению, не видно на чернобелом рисунке). Для построения спектров используется прямое преобразование Фурье – функция fft.

Этот пример наглядно иллюстрирует технику фильтрации. Рекомендуется просмотреть дополнительные примеры, которые есть в разделе **Demos** системы применительно к пакету расширения **Signal Processing** (если этот пакет установлен).

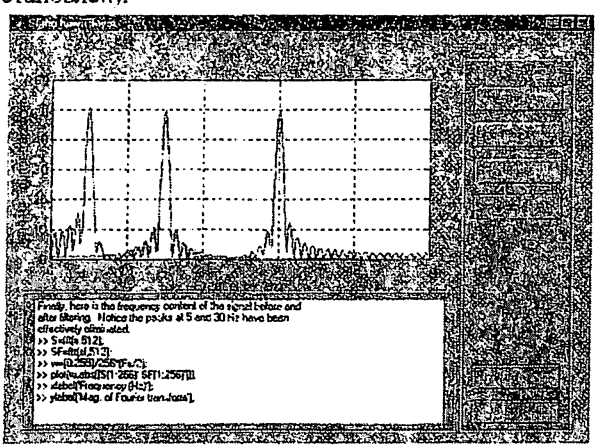


Рисунок 5.4 - Анализ спектров сигналов на входе и на выходе фильтра и построение их спектров

Для осуществления двумерной фильтрации служит функция filter2:

- *filter2(B,X)* фильтрует данные в двумерном массиве X, используя дискретный фильтр, описанный матрицей B. Результат Y имеет те же размеры, что и X;
- *filter2(B,X,'option')* выполняет то же, но с опцией, влияющей на размер массива Y:
  - 'same' – size(Y)=size(X) (действует по умолчанию);
  - 'valid' – size(Y) < size(X), центральная часть двумерной свертки, при вычислении которой не приходится дополнять массивы нулями;
  - 'full' – size(Y) > size(X), полная двумерная свертка.

Входные параметры.

`b` — Коэффициенты числителя рациональной передаточной функции в виде вектора.

Типы данных:

double | `single` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical`

Поддержка комплексного числа: Да

`a` — Коэффициенты знаменателя рациональной передаточной функции в виде вектора.

Типы данных:

double | `single` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical` **Поддержка комплексного числа: Да**

`x` — Входные данные в виде вектора, матрицы или многомерного массива.

Типы данных:

double | `single` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical` **Поддержка комплексного числа: Да**

`zi` — Начальные условия для задержек фильтра `[]` (значение по умолчанию) | вектор | матрица | многомерный массив

Начальные условия для фильтра задаются в виде вектора, матрицы или многомерного массива.

- Если zi вектор, затем его длиной должен быть max(length(a),length(b))-1.
- Если zi матричный или многомерный массив, затем размером ведущей размерности должен быть max(length(a),length(b))-1. Размер каждого остального измерения должен совпадать с размером соответствующей размерности x. Например, рассмотрите использование filter вдоль второго измерения (dim = 2) из 3 4 5 массивами x. Массив zi должен иметь размер [max(length(a),length(b))-1]-3-на-5.

Значение по умолчанию, заданное [], инициализирует все задержки фильтра с нулем.

Типы данных:

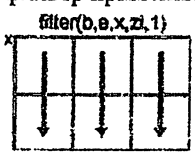
double | `single` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logic al` **Поддержка комплексного числа: Да**

`dim` — Размерность, которая задает направление расчета положительный целочисленный скаляр

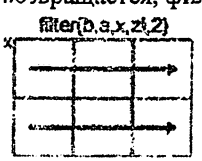
Величина для работы, заданная как положительный целый скаляр. Если значение не задано, то по умолчанию это 1 размер массива, не равный 1.

Рассмотрите двумерный входной массив, x.

- Если dim = 1, затем filter(b,a,x,zi,1)работает вдоль строк x и возвращается, фильтр применился к каждому столбцу.



- Если dim = 2, затем filter(b,a,x,zi,2) работает вдоль столбцов x и возвращается, фильтр применился к каждой строке.



Если dim больше ndims(x), затем filter возвращает x.

Типы данных:

double | `single` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logic al`

Выходные аргументы.

`y` — Фильтрованные данные вектор | матрица | многомерный массив

Фильтрованные данные, возвращенные как вектор, матрица или многомерный массив одного размера с входными данными, x.

Если x имеет тип single, затем filter исходно вычисляет в одинарной точности и y также имеет тип single. В противном случае, y возвращен как тип double.

Типы данных: double | single

`zf` — Итоговые условия для задержек фильтра вектор | матрица | многомерный массив

Итоговые условия для задержек фильтра, возвращенных как вектор, матрица или многомерный массив.

- Если x вектор, затем zf вектор-столбец длины max(length(a),length(b))-1.
- Если x матричный или многомерный массив, затем zf массив вектор-столбцов длины max(length(a),length(b))-1, таким образом, что количество столбцов в zf эквивалентно количеству столбцов в x. Например, рассмотрите использование filter вдоль второго измерения (dim = 2) из 3 4 5 массивами x. Массив zf имеет размер [max(length(a),length(b))-1]-3-на-5.

Типы данных: double | single

Пример выполнения практической работы № 5

Фильтр скользящего среднего значения.

Фильтр скользящего среднего значения является общепринятой методикой, используемой для сглаживания зашумленных данных. Этот пример использует filter функция, чтобы вычислить средние значения вдоль вектора из данных.

Создайте 1 100 вектор-строку из синусоидальных данных, которые повреждаются случайным шумом.

```
t = linspace (-pi, pi, 100);
rng default  % инициализировать генератор случайных чисел
x = sin (t) + 0,25 * rand (size (t));
```

Фильтр скользящего среднего значения двигает окно длины $WindowSize$ вдоль данных вычислительные средние значения данных содержатся в каждом окне. Следующее разностное уравнение задает фильтр скользящего среднего значения вектора *Икс*:

$$y(n) = \frac{1}{WindowSize}(x(n) + x(n-1) + \ldots + x(n-(windowSize-1)).$$

Для размера окна 5, вычислите числитель и коэффициенты знаменателя для рациональной передаточной функции.

```
windowSize = 5;
b = (1/windowSize)*ones(1,windowSize);
a = 1;
```

Найдите скользящее среднее значение данных и постройте его против исходных данных.

```
y = filter(b,a,x);

plot(t,x)
hold on
plot(t,y)
legend('Input Data','Filtered Data')
```

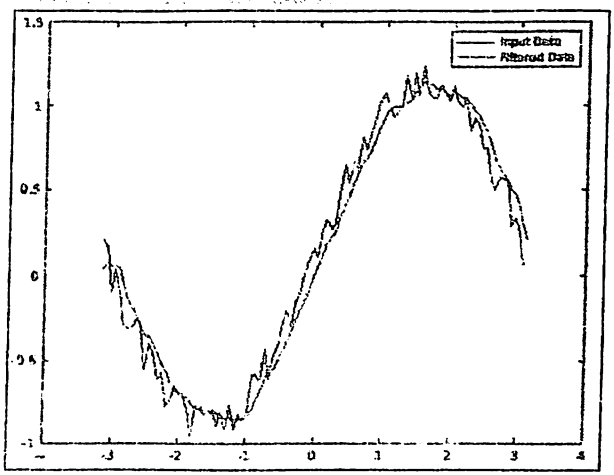


Рисунок 5.5 - График выходных и отфильтрованных данных. Пример 1.

```
y = filter(b,a,x);

t = 1:length(x),
plot(t,x,'+-',t,y,'.-')
legend('Original Data','Filtered Data')
```

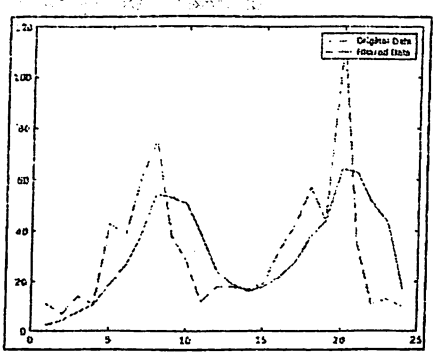


Рисунок 5.6 - График выходных и отфильтрованных данных. Пример 2.

Фильтрация матричных строк

Этот пример фильтрует матрицу данных со следующей рациональной передаточной функцией.

$$H(z) = \frac{b(1)}{a(1) + a(2)z^{-1}} = \frac{1}{1 - 0.2z^{-1}}$$

Создайте 2 15 матрицу случайных входных данных.

```
rng default  % инициализировать генератор случайных чисел
x = rand (2,15);
```

Задайте числитель и коэффициенты знаменателя для рациональной передаточной функции.

```
b = 1;
a = [1 -0,2];
```

Примените передаточную функцию вдоль второго измерения x и возвратите 1D цифровой фильтр каждой строки. Постройте первую строку исходных данных против отфильтрованных данных.

```
y = filter(b,a,x,[],2);

t = 0:length(x)-1;  %index vector

plot(t,x(1,:))
hold on
plot(t,y(1,:))
legend('Input Data','Filtered Data')
title('First Row')
```

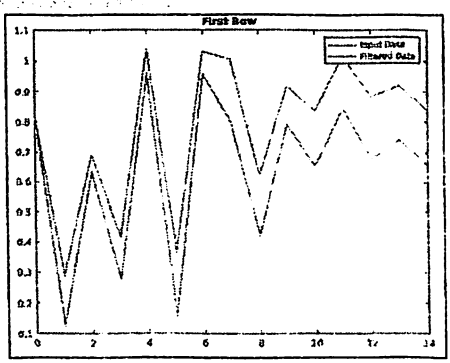


Рисунок 5.6 – График первой строки выходных данных и отфильтрованных данных

Построим вторую строку входных данных и сравним с отфильтрованными данными.

```
figure
plot(t,x(2,:))
hold on
plot(t,y(2,:))
legend('Input Data','Filtered Data')
title('Second Row')
```

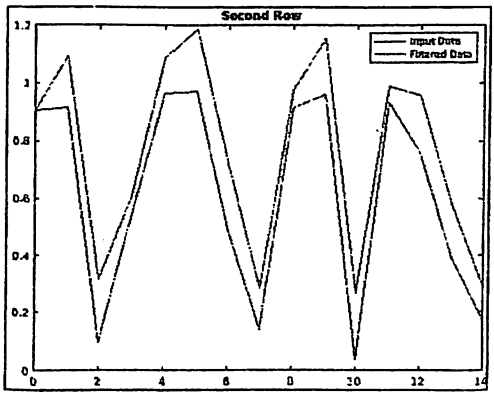


Рисунок 5.7 – График первой строки выходных данных и отфильтрованных данных

Задание для выполнения практической работы № 5

Задание 1. Построить график произвольной тригонометрической функции по примеру указанному в рисунке 5.1. Вывести на экран график функции фильтра (рис. 5.2). Задать функции для фильтра и вывести график функции фильтра на экран. Необходимо вывести отфильтрованный график на экран (рис. 5.3).

Задание 2. Построить график используя фильтр скользящего среднего значения, используемой для сглаживания зашумленных данных.

Задание 3. Фильтрация матричных строк. Необходимо произвести фильтрацию матричных строк, используя рациональную. Передаточную функцию.

Содержание и оформление отчета:

1. Наличие титульного листа, шрифт Times New Roman – 14, интервал 1,5;
2. Выполнение всех приложенных заданий;
3. Заключение по выполненной работе;
4. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Какие операции выполняется при анализ данных в среде МАТЛАБ?
2. Какие операции выполняется при фильтрация данных в среде МАТЛАБ?
3. Какие входные параметры имеется в среде МАТЛАБ?
4. Какие выходные параметры имеется в среде МАТЛАБ?

Практическая работа № 6
СОЗДАНИЕ МОДЕЛИ ВЫХОДНОЙ РЕГРЕССИИ С ОДНОЙ ПЕРЕМЕННОЙ MATLAB (ЛИНЕЙНАЯ РЕГРЕССИЯ)

Цель работы: Создание модели линейной регрессии в среде MATLAB.

Линейная регрессия соответствует модели данных, которая линейна в коэффициентах модели. *Модель данных* явным образом описывает отношение между переменными *predictor* и *response*. Наиболее распространенным типом линейной регрессии является *least-squares fit*, который может соответствовать и линиям и полиномам среди других линейных моделей.

Перед моделированием связи между парами величин рекомендуется провести корреляционный анализ, чтобы установить, существует ли линейная связь между этими величинами. Следует иметь в виду, что переменные могут иметь нелинейные отношения, которые не может обнаружить корреляционный анализ. Для получения дополнительной информации смотрите Линейную корреляцию.

MATLAB® Пользовательский интерфейс Basic Fitting помогает вам соответствовать своим данным, таким образом, можно вычислить коэффициенты модели и построить модель сверху данных. Для примера смотрите Пример: Используя Пользовательский интерфейс Basic Fitting. Также можно использовать MATLAB polyfit и polyval функции, чтобы соответствовать вашим данным к модели, которая линейна в коэффициентах. Для примера см. Программное аппроксимирование.

Если необходимо соответствовать данным нелинейной моделью, преобразуйте переменные, чтобы сделать отношение линейным. В качестве альтернативы попытайтесь соответствовать нелинейной функции непосредственно с помощью любого Statistics and Machine Learning Toolbox™ nlinfit функция, Optimization Toolbox™ lsqcurvefit функция, или путем применения функций в Curve Fitting Toolbox™.

В этом примере показано, как выполнить простую линейную регрессию с помощью accidents набор данных.

Линейная регрессия моделирует отношение между зависимым, или ответ, переменную y и один или несколько независимый, или предиктор, переменные $x_1,...,x_n$.

Простая линейная регрессия рассматривает только одну независимую переменную с помощью отношения

$$y=\beta_0+\beta_1 x+\epsilon,$$

где

- y_i— зависимая переменная (отклик)
- x_i— известная константа (значение объясняющей переменной, измерянной в i-ом эксперименте)
- β_0, β_1 — параметры модели (свободный член и угловой коэффициент).
- ϵ_i — случайная ошибка

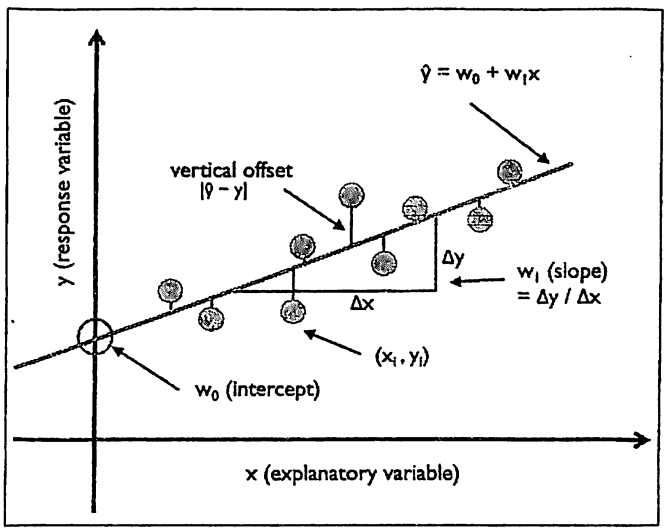


Рисунок 6.1 – График линейной регрессии

Начните с ряда n наблюдаемые величины x и y данный $(x1,y1)$, $(x2,y2)$, ..., (x_n,y_n). Используя простое отношение линейной регрессии, эти значения формируют систему линейных уравнений. Представляйте эти уравнения в матричной форме как

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_i \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$

Пусть

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, B = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$

Отношение теперь *Y=XB*.

В MATLAB можно найти *B* использование mldivide оператор как B = X\Y.

Примеры создания значений x и y:

- **Способ 1:**

```
>> x = linspace(0,10,15)';
>> y = linspace(10,20,15)';
```

- **Способ 2:**

```
>> x = rand(15, 1);
>> y = rand(15, 1);
```

Или

```
>> y =sin(x);
```

- **Способ 3:**

```
>> x =(-3:0.2:3)';
>> y =(-2:0.2:2)';
```

- **Способ 4:**

```
>> x = rand(10,1,1);
>> y = rand(10,1,1);
```

- **Способ 5:**

```
>> x0 = [2,2,2,4,4,6,6,6,10,10];
>> x = x0';
>> y0 = [1,2,3,4,5,6,7,8,9,10];
>> y = y0';
```

Так же, в каждом из способов, значение «y» можно задать функцией зависимости:

```
>>y = x + 1.5*sin(x) + randn(size(x,2),1)
```

Этапы выполнения практической работы:

```
%% Смоделировать исходные данные
% Если файл данных отстутствует, можно создать данные
самостятельно

x = linspace(0,10,200)'; % независимая переменная,
значения
y = x + 1.5*sin(x) + randn(size(x,1),1); % зависимая
переменная имеет наложенную случайную переменную с
нулевым матожиданием и однородной дисперсией

%% Одномерная регрессия
```

```
% Всесстанавливаемая регрессионная зависимость - прямая
на плоскости.
model1 = 'y=w_1+w_2x';
A = [x.^0, x];              % построить матрицу подстановок
                            % x - (m,1)-вектор, y - (m,1)-
вектор
b = (A'*A)\(A'*y);          % решить нормальное уравнение
                            % методом гауссова исключения
b = pinv(A'*A)*(A'*y);% вариант обращения матрицы
y1 = b(2)*x;
y2 = b(1)+b(2)*x;           % восстановить зависимую
переменную
                            % при заданных значениях x
r = y-y1;                   % найти вектор регрессионных
остатков
SSE = r'*r                  % подсчитать ошибку
% нарисовать график
plot (x,y, 'o');
hold on
plot (x,y1); % (график ниже, на картинке)
plot (x,y2);
```

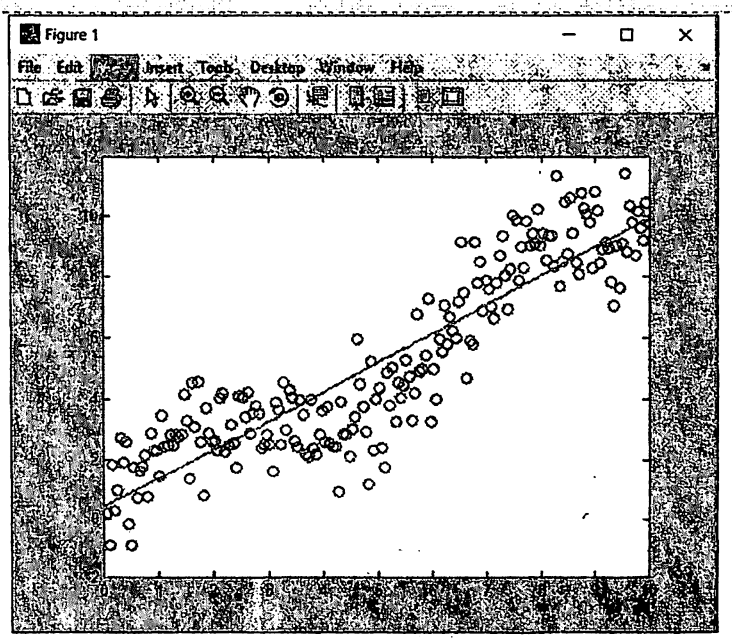


Рисунок 6.2 – Результат выполненный работы.

Пример выполнения практической работы № 6

Часть 1.

```
>> x0 = [2,2,2,4,4,6,6,6,10,10];
>> x = x0';
(Нужно вывести значения x)
>> y0 = [1,2,3,4,5,6,7,8,9,10];
>> y = y0';
(Нужно вывести значения y)
>> format long
>> b = x\y;
>> y1 = b*x;
(Нужно вывести значения y1)
>> plot(x,y,'o')
```

Или

```
>> scatter(x,y)
>> hold on
>> plot(x,y1, 'r')
```

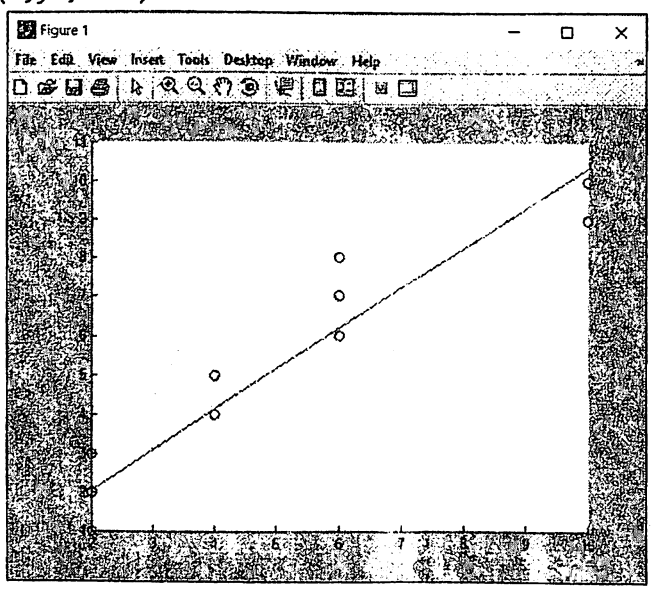


Рисунок 6.3 – График линейной регрессии.

Настроим вид полученного графика:

```
>> xlabel('Nazvaniy osi x')
```

```
>> ylabel('Nazvaniy osi y')
>> title('Lineynaya Regressiya')
>> grid on
```

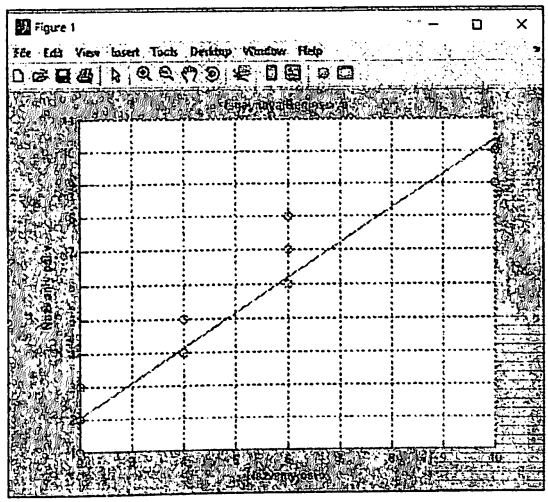


Рисунок 6.4 – Настроенный график линейной регрессии.

Улучшаем подгонку включением свободного члена β_0 в вашей модели как $y=\beta_0+\beta_1 x$. Вычислить β_0 путем дополнения x со столбцом из единиц и использования \ оператор.

Часть 2.

Варианта № 1:

```
>> X = [ones(length(x),1) x];
>> b1 = X\y
b1 =

   0.529411764705884
   0.955882352941177

>> y2 = X*b1;
>> plot(x,y2,'--')
>>legend('Data','Slope','Slope & Intercept','Location','best');
```

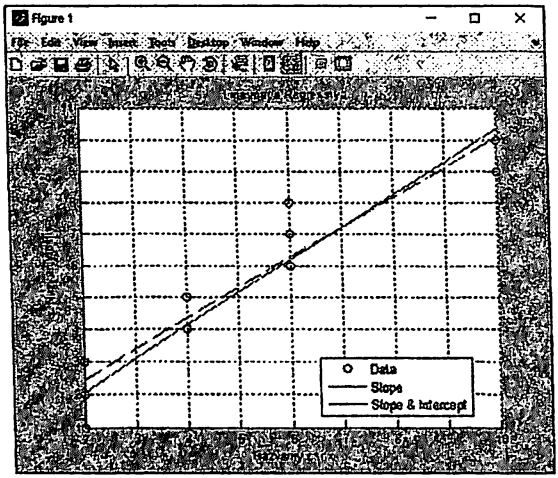


Рисунок 6.5 – Улучшенная подгонка графика линейной регрессии. Вариант 1

Вариант № 2

```
>> A = [x.^0, x];
>> b1 = (A'*A)\(A'*y);
>> b1 = pinv(A'*A)*(A'*y)
b1 =
   0.529411764705884
   0.955882352941177
>> y2 = b1(1)+b1(2)*x;
>> plot(x,y2,'--')
```

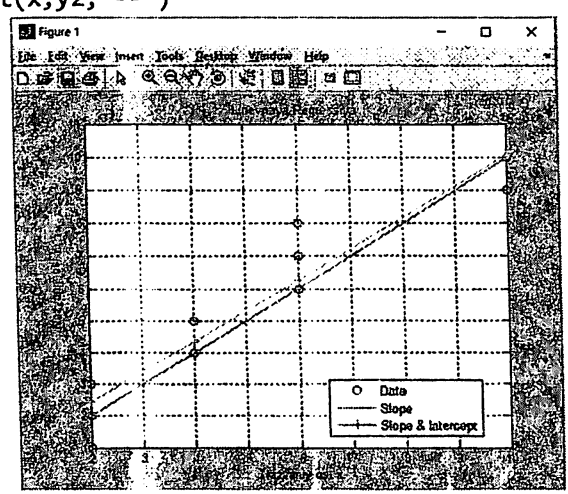


Рисунок 6.5 – Улучшенная подгонка графика линейной регрессии. Вариант 2

Задание для выполнения практической работы № 6

Задание 1. По варианту выбрать способ объявления значения x и y, после объявления вывести значения на экран. Далее необходимо вычислить коэффициент регрессии «b», затем отношение линейной регрессии «y1».

Задание 2. По полученным данным построить график y(x) и график y1(x). Вывести оба графика на одном экране, первым в виде точек (scatter) на оси координат, второй в виде линии.

Задание 3. Улучшить подгонку включением свободного члена β, вывести полученный график на тот же экран. Настроить вид полученного графика, добавить название осей (x и y) и название графика (в виде фамилии и номера группы).

Содержание и оформление отчета:

1. Наличие титульного листа, шрифт Times New Roman – 14, интервал 1,5;
2. Выполнение всех приложенных заданий;
3. Заключение по выполненной работе;
4. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Что такое линейная регрессия в среде МАТЛАБ?
2. Какие выходные модели используется с одной переменной в среде МАТЛАБ?
3. Что такое модель данных в среде МАТЛАБ?
4. Какие функции используется в модели данных в среде МАТЛАБ?

Практическая работа № 7
МНОГОВАРИАНТНОЕ РЕГРЕССИОННОЕ МОДЕЛИРОВАНИЕ ВЫВОДА В MATLAB (МНОЖЕСТВЕННАЯ РЕГРЕССИЯ)

Цель работы: Создание модели множественной регрессии в среде MATLAB.

Регрессионный анализ (англ.: Regression Analysis) - это метод статистического анализа данных, цель которого состоит в том, чтобы понять, связаны ли две или более переменные, связаны ли направление и сила, и создать математическую модель для наблюдения конкретных переменных, чтобы предсказать исследователей Переменные интереса. В частности, регрессионный анализ может помочь людям понять, насколько сильно изменяется зависимая переменная, когда изменяется только одна независимая переменная. В целом, с помощью регрессионного анализа мы можем оценить условное ожидание зависимой переменной от заданных независимых переменных.

Регрессионный анализ должен установить связь между зависимой переменной Y Y (или зависимой переменной, анти-зависимой переменной) и независимой переменной X (или независимой переменной, пояснительной переменной). Простая линейная регрессия использует одну независимую переменную X, а комплексная регрессия использует более одной независимой переменной (X1,X2...Xi X 1 , X 2 . . . X i)

Метод регрессионного анализа относится к использованию принципов статистики данных для математической обработки большого количества статистических данных, определения корреляции между зависимой переменной и некоторыми независимыми переменными, установления уравнения регрессии (функционального выражения) с хорошей корреляцией и добавления экстраполяции. аналитический метод, используемый для прогнозирования будущих изменений зависимых переменных. По количеству зависимых и независимых переменных его можно разделить на: унарный регрессионный анализ и множественный регрессионный анализ; по функциональному выражению зависимой переменной и независимой переменной его можно разделить на: линейный регрессионный анализ и нелинейный регрессионный анализ.

Множественная линейная регрессия

Для простоты примем, что x_0 = 1x0=1. Тогда в общем случае форма линейной регрессии выглядит следующим так:

$$\forall h \in \mathcal{H}, h\left(\vec{x}\right) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m = \sum_{i=0}^{m} w_i x_i = \vec{x}^T \vec{w}$$

Пример графика для линейной регрессионной модели с двумя независимыми переменными:

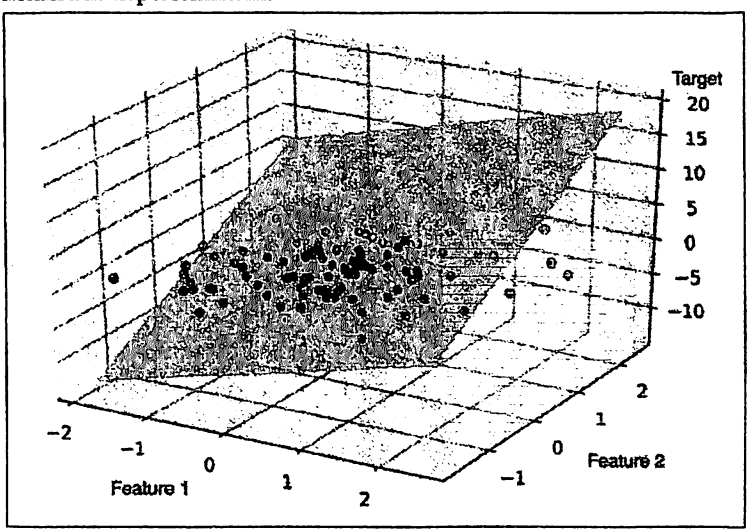


Рисунок 7.1 – Пример линейной регрессионной модели с двумя независимыми переменными

Классификация:

1. Классификация по количеству зависимых и независимых переменных: унарный регрессионный анализ и множественный регрессионный анализ;
2. Классификация по функциональному выражению зависимой переменной и независимой переменной: линейный регрессионный анализ и нелинейный регрессионный анализ.

Основные решаемые проблемы:

1. Определить, есть ли корреляция между переменными, и если да, найдите математическое выражение;
2. По значению одной или нескольких переменных прогнозируйте или контролируйте значение другой или нескольких переменных, и необходимо оценить точность такого контроля или прогноза.

Этапы регрессионного анализа:

1. Предварительно установите уравнение регрессии на основе имеющихся данных и взаимосвязи между независимой переменной и зависимой переменной;
2. Найдите разумный коэффициент регрессии;
3. Выполните тест корреляции, чтобы определить коэффициент корреляции;
4. После выполнения требований релевантности вы можете определить будущее положение вещей на основе комбинации полученного уравнения регрессии и конкретных условий и рассчитать доверительный интервал прогнозируемого значения.

Срок действия и меры предосторожности

Эффективность: Прогнозы с регрессионным анализом должны сначала делать прогнозы для каждой независимой переменной. Если каждой независимой переменной можно управлять вручную или ее легко предсказать, а уравнение регрессии более реалистично, применение прогнозирования регрессии будет эффективным, в противном случае его будет сложно применить;

Меры предосторожности: Чтобы сделать уравнение регрессии более реалистичным, мы должны, во-первых, определить возможные типы и количество независимых переменных, насколько это возможно, и качественно определить возможные типы уравнения регрессии на основе соблюдения закона развития вещей; во-вторых, стремиться получить достаточно качественные статистические данные, затем использовать статистические методы, использовать математические инструменты и соответствующее программное обеспечение для расчета или улучшения качественных суждений с количественной точки зрения.

Команда регресса Используется для унарной и множественной линейной регрессии, по сути, метода наименьших квадратов. Введите help regress в командной строке matlab, чтобы получить информацию, связанную с командой:

```
regress - Multiple linear regression

    This MATLAB function returns a p-by-1 vector b of
coefficient estimates for a
    multilinear regression of the responses in y on the
predictors in X.
b = regress(y,X)
```

```
[b,bint] = regress(y,X)
[b,bint,r] = regress(y,X)
[b,bint,r,rint] = regress(y,X)
[b,bint,r,rint,stats] = regress(y,X)
[...] = regress(y,X,alpha)
```

Объяснение параметра:

B: Коэффициент регрессии - это вектор («вектор B коэффициентов регрессии в линейной модели Y = X * B»).

BINT: интервальная оценка коэффициента регрессии («матрица BINT из 95% confidence intervals for B")

R: Остаток («вектор R остатков»).

RINT: Доверительный интервал («матрица RINT интервалов, которая может использоваться для диагностики выбросов»).

СТАТИСТИКА: статистика, используемая для тестирования регрессионных моделей. Иметь4Числовые значения: коэффициент детерминации R ^2, F статистическое значение наблюдения, значение теста p, оценка дисперсии ошибки.

АЛЬФА: Уровень значимости (значение по умолчанию при отсутствии0.05

Пример выполнения практической работы № 7

Целевая функция: $y = Ax_1^2 + Bx_1^2 + Cx_1 + Dx_2 + Ex_1 \cdot x_2 + F$ (это квадратичная функция, две переменные, прописные буквы являются константами)

Образец кода:

```
% Целевая функция: y = Ax1 ^ 2 + Bx2 ^ 2 + Cx1 + Dx2 + Ex1 *
x2 + F (это квадратичная функция, две переменные, прописные
буквы являются константами)
% Импорт данных
    y=[7613.51   7850.91   8381.86   9142.81 10813.6 8631.43
8124.94  9429.79  10230.81  10163.61  9737.56  8561.06  7781.82
7110.97]';
    x1=[7666 7704 8148 8571 8679 7704 6471 5870 5289 3815
3335 2927 2758 2591]';
    x2=[16.22 16.85 17.93 17.28 17.23 17 19 18.22 16.3 13.37
11.62 10.36 9.83 9.25]';
X=[ones(size(y)) x1.^2 x2.^2 x1 x2 x1.*x2];
```

```
% Начать анализ
[b,bint,r,rint,stats] = regress(y,X);
```

Результат вывода:

```
b =

   1.0e+04 *
  -1.353935450267785
   0.000000089381408
  -0.005811190715467
  -0.000605427789545
   0.479983626458515
  -0.000037869040292

bint =

   1.0e+04 *
  -2.621944842897244  -0.085926057638326
   0.000000034253753   0.000000144509063
  -0.027588831662545   0.015966450231610
  -0.001309493882546   0.000098638303455
   0.119564693553895   0.840402559363136
  -0.000105954336341   0.000030216255756

r =

   1.0e+02 *
  -4.397667358983981
  -2.361417286008691
  -1.434643909138104
  -5.904203974279499
   7.511701773845143
   5.570806699070745
  -2.447861341816488
   0.494622057475135
   6.376995507987686
  -6.789520765534580
   2.744335484633575
   1.578124015815665
  -0.803533566911919
  -0.137737336155578

rint =

   1.0e+03 *
```

```
  -1.200286938109502   0.320753466312705
  -1.396750443678483   0.924466986476745
  -0.899861317279054   0.612932535451433
  -1.544523004301250   0.363682209445351
   0.127290215500605   1.375050139268424
  -0.505983092518450   1.620144432332599
  -1.145989115179227   0.656416846815929
  -0.952086756810051   1.051011168305078
  -0.372857973848901   1.648257075446438
  -1.420201937940137   0.062297784833221
  -0.889572649262705   1.438439746189421
  -1.057885008292926   1.373509811456059
  -1.241713948436468   1.081007235054084
  -0.992490338296236   0.964942871065120

stats =

   1.0e+05 *

   0.000008444011951          0.000086828553270          0.000000043344434
3.162249735298973
```

b - соответствующий параметр, b (1) - это F (последний постоянный член), b (2) - это A (первый параметр), b (3) - это B, b (4) - это C, b (4) - это D, b (5) - это E. Bint - это 95% доверительный интервал b. Третий параметр статистики - это значение P теста F. Значение p очень мало (P <0,001), что указывает на эффективность модели подгонки.

```
scatter3(x1,x2,y,'filled') % разброса можно использовать для построения графиков разброса
```

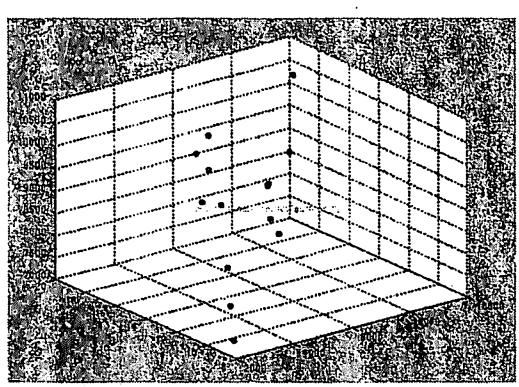


Рисунок 7.2 - График разброса выходных данных

Приближенное трехмерное изображение:

```
% Отображение 3D вида
    hold on  % Не рассчитывайте данные четко и продолжайте рисовать на диаграмме рассеяния прямо сейчас
    x1fit = min(x1):100:max(x1);     % Установите интервал данных x1
    x2fit = min(x2):1:max(x2);       % Установите интервал данных x2
    [X1FIT,X2FIT] = meshgrid(x1fit,x2fit);     % Создает двумерную плоскость сетки, можно также сказать, генерирует координаты X1FIT, X2FIT
    YFIT=b(1)+b(2)*X1FIT.^2+b(3)*X2FIT.^2+b(4)*X1FIT+b(5)*X2FIT+b(6)*X1FIT.*X2FIT;     % Подставьте полученные параметры, чтобы они соответствовали формуле функции
    mesh(X1FIT,X2FIT,YFIT)       % X1FIT, X2FIT - матрица координат сетки, YFIT - матрица высоты в точке сетки
    view(10,10)  % Измените угол для просмотра существующего трехмерного изображения, первые 10 представляют азимутальный угол, а вторые - угол обзора сверху.
                 % Азимутальный угол эквивалентен долготе в сферических координатах, а угол обзора сверху эквивалентен широте в сферических координатах.
    xlabel('x1') % Задайте имя оси X
    ylabel('x2') % Задайте имя оси Y
    zlabel('y')  % Задайте имя оси z
```

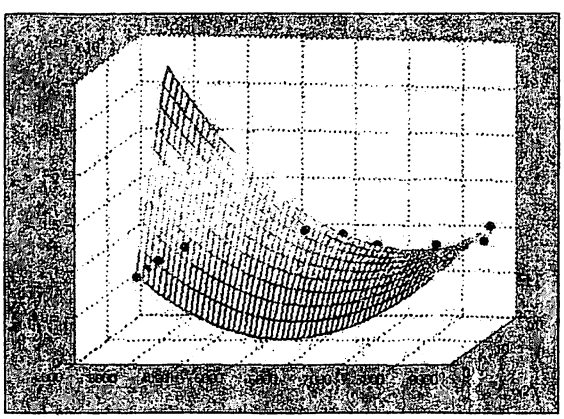


Рисунок 7.2 – Трехмерное изображение эффекта подгонки

Задание для выполнения практической работы № 7

Задание 1. Необходимо задать значения y, x1 и x2, вручную или с помощью функции rand (x2=rand(14,1)*100).

Задание 2. Задать целевую функцию, как показано в примере. Вывести значения «b», «bint», «r», «rint», «stats»

Задание 3. Построить график разброса данных по введенным данным, как показано на рисунке 7.1. Построить приближенное трехмерное изображение как показано на рисунке 7.2

Содержание и оформление отчета:

1. Наличие титульного листа, шрифт Times New Roman – 14, интервал 1,5;
2. Выполнение всех приложенных заданий;
3. Заключение по выполненной работе;
4. Ответы на контрольные вопросы.

Контрольные вопросы

1. Что такое регрессионный анализ?
2. Опишите матричный подход к определению коэффициента регрессии.
3. Опишите этапы регрессионного анализа
4. Дайте классификацию регрессионного анализа.
5. Приведите форму множественной линейной регрессии.

Практическая работа № 8
КЛАССИФИКАЦИЯ ДАННЫХ SVM В MATLAB

Цель работы: Провести классификацию данных SVM в среде MATLAB.

Метод опорных векторов

Для начала вы обучаете машину опорных векторов, и затем, используйте обученную машину, чтобы классифицировать (предсказывают) новые данные. Кроме того, чтобы получить удовлетворительную прогнозирующую точность, можно использовать различные функции ядра SVM, и необходимо настроить параметры функций ядра. Для этого необходимо сделать:

- Обучение классификатора SVM
- Классификация новых данных с классификатором SVM
- Настройка классификатора SVM

Изучение классификатора SVM

Использование классификатора SVM fitcsvm имеет наиболее распространенный синтаксис:

```
SVMModel = fitcsvm(X,Y,'KernelFunction','rbf',...
    'Standardize',true,'ClassNames',{'negClass','posClass'});
```

Входные параметры:

- X — Матрица данных о предикторе, где каждая строка является одним наблюдением и каждым столбцом, является одним предиктором.
- Y — Массив класса помечает каждой строкой, соответствующей значению соответствующей строки в XY может быть категориальное, символ, или массив строк, логический или числовой вектор или массив ячеек из символьных векторов.
- KernelFunction — Значением по умолчанию является 'linear' для изучения 2D класса, которое разделяет данные гиперплоскостью. Значение 'gaussian' (или 'rbf') значение по умолчанию для изучения одного класса и задает, чтобы использовать Гауссово (или радиальная основная функция) ядро. Важный шаг, чтобы успешно обучить классификатор SVM должен выбрать соответствующую функцию ядра.
- Standardize — Отметьте указание, должно ли программное обеспечение стандартизировать предикторы перед обучением классификатор.
- ClassNames — Различает отрицательные и положительные классы или задает который классы включать в данные. Отрицательный класс является первым элементом (или строка символьного массива), например, 'negClass', и

положительный класс является вторым элементом (или строка символьного массива), например, 'posClass'. ClassNames должен быть совпадающий тип данных как Y. Это - хорошая практика, чтобы задать имена классов, особенно если вы сравниваете эффективность различных классификаторов.

Получившаяся, обученная модель (SVMModel) содержит оптимизированные параметры из алгоритма SVM, позволяя вам классифицировать новые данные.

Для большего количества пар "имя-значение" можно использовать, чтобы управлять обучением, видеть fitcsvm страница с описанием.

Классификация новых данных с классификатором SVM

Классифицируйте новые данные с помощью predict. Синтаксис для классификации новых данных с помощью обученного классификатора SVM (SVMModel):

```
[label,score] = predict(SVMModel,newX);
```

Итоговый вектор, label, представляет классификацию каждой строки в X. score *n*-by-2 матрица мягких баллов. Каждая строка соответствует строке в X, который является новым наблюдением. Первый столбец содержит оценки для наблюдений, классифицируемых на отрицательный класс, и второй столбец содержит наблюдения баллов, классифицируемые на положительный класс..

Настройка классификатора SVM

Используйте 'OptimizeHyperparameters' аргумент пары "имя-значение" fitcsvm найти значения параметров, которые минимизируют потерю перекрестной проверки. Имеющими право параметрами является 'BoxConstraint', 'KernelFunction', 'KernelScale', 'PolynomialOrder', и 'Standardize'. Для примера смотрите, Оптимизируют Подгонку Классификатора Используя Байесовую Оптимизацию. В качестве альтернативы можно использовать bayesopt функция, как показано в Оптимизируют перекрестный Подтвержденный Классификатор Используя bayesopt. bayesopt функция позволяет большей гибкости настраивать оптимизацию. Можно использовать bayesopt функция, чтобы оптимизировать любые параметры, включая параметры, которые не имеют право оптимизировать, когда вы используете fitcsvm функция.

Можно также попытаться настроить параметры классификатора вручную согласно этой схеме:

1. Передайте данные fitcsvm, и набор аргумент пары "имя-значение" 'KernelScale','auto'. Предположим, что обученная модель SVM называется SVMModel. Программное обеспечение использует

эвристическую процедуру, чтобы выбрать шкалу ядра. Эвристическая процедура использует подвыборку. Поэтому, чтобы воспроизвести результаты, установите seed случайных чисел с помощью rng перед обучением классификатор.

2. Крест подтверждает классификатор путем передачи его crossval. По умолчанию программное обеспечение проводит 10-кратную перекрестную проверку.
3. Передайте перекрестную подтвержденную модель SVM kfoldLoss оценить и сохранить ошибку классификации.
4. Переобучите классификатор SVM, но настройте 'KernelScale' и 'BoxConstraint' аргументы в виде пар имя-значение.

- BoxConstraint — Одна стратегия состоит в том, чтобы попробовать геометрическую последовательность параметра ограничения поля. Например, примите 11 значений от 1e-5 к 1e5 на коэффициент 10. Увеличение BoxConstraint может сократить число векторов поддержки, но также и может увеличить учебное время.
- KernelScale — Одна стратегия состоит в том, чтобы попробовать геометрическую последовательность параметра сигмы RBF, масштабируемого в исходной шкале ядра. Сделайте это:
  a. Получая исходную шкалу ядра, например, ks, использование записи через точку: ks = SVMModel.KernelParameters.Scale.
  b. Используйте в качестве новых факторов шкал ядра оригинала. Например, умножьте ks этими 11 значениями 1e-5 к 1e5, увеличение на коэффициент 10.

Выберите модель, которая дает к самой низкой ошибке классификации. Вы можете хотеть далее совершенствовать свои параметры, чтобы получить лучшую точность. Начните со своих начальных параметров и выполните другой шаг перекрестной проверки, на этот раз с помощью фактора 1,2.

Оптимизация подгонки классификатора, используя байесовую оптимизацию

В этом примере показано, как оптимизировать классификацию SVM с помощью `fitcsvm` функцииn, где `OptimizeHyperparameters` аргумент значения имени.

Генерация данных

Классификация работает над местоположениями точек от смешанной гауссовской модели. В *Элементах Статистического Изучения*, Hastie,

Тибширэни и Фридмана (2009), страница 17 описывает модель. Модель начинается с генерации 10 базисных точек для "зеленого" класса, распределенного как 2D независимые нормали со средним значением (1,0) и модульное отклонение. Это также генерирует 10 базисных точек для "красного" класса, распределенного как 2D независимые нормали со средним значением (0,1) и модульное отклонение. Для каждого класса (зеленый и красный), сгенерируйте 100 случайных точек можно следующим образом:

1. Выберите базисную точку *m* соответствующего цвета однородно наугад.
2. Сгенерируйте независимую случайную точку с 2D нормальным распределением со средним значением *m* и отклонением I/5, где я - единичная матрица 2 на 2. В этом примере используйте отклонение I/50, чтобы показать преимущество оптимизации более ясно.

Сгенерируйте эти 10 базисных точек для каждого класса.

```
rng('default') % For reproducibility
grnpop = mvnrnd([1,0],eye(2),10);
redpop = mvnrnd([0,1],eye(2),10);
Просмотрите базисные точки.
plot(grnpop(:,1),grnpop(:,2),'go')
hold on
plot(redpop(:,1),redpop(:,2),'ro')
hold off
```

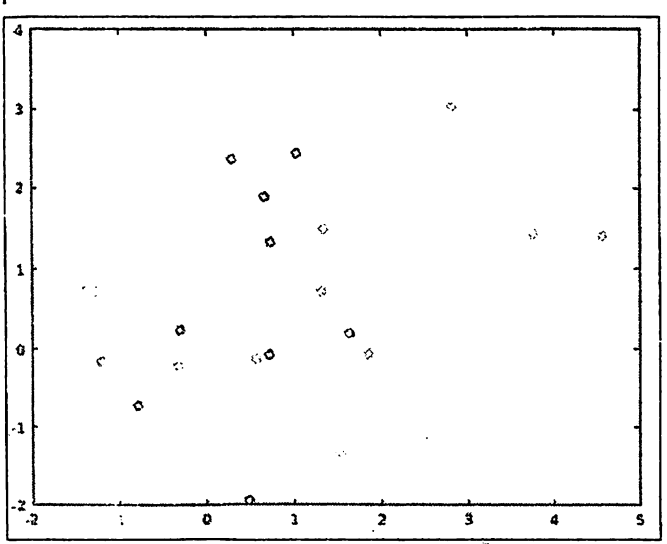


Рисунок 8.1 – 10 базисных точек

Поскольку некоторые красные базисные точки близко к зеленым базисным точкам, это может затруднить классификацию точки данных на основе одного только местоположения.

Сгенерируйте 100 точек данных каждого класса.

```
redpts = zeros(100,2);
grnpts = redpts;
for i = 1:100
    grnpts(i,:) = mvnrnd(grnpop(randi(10),:),eye(2)*0.02);
    redpts(i,:) = mvnrnd(redpop(randi(10),:),eye(2)*0.02);
end
```

Просмотрите точки данных.

```
figure
plot(grnpts(:,1),grnpts(:,2),'go')
hold on
plot(redpts(:,1),redpts(:,2),'ro')
hold off
```

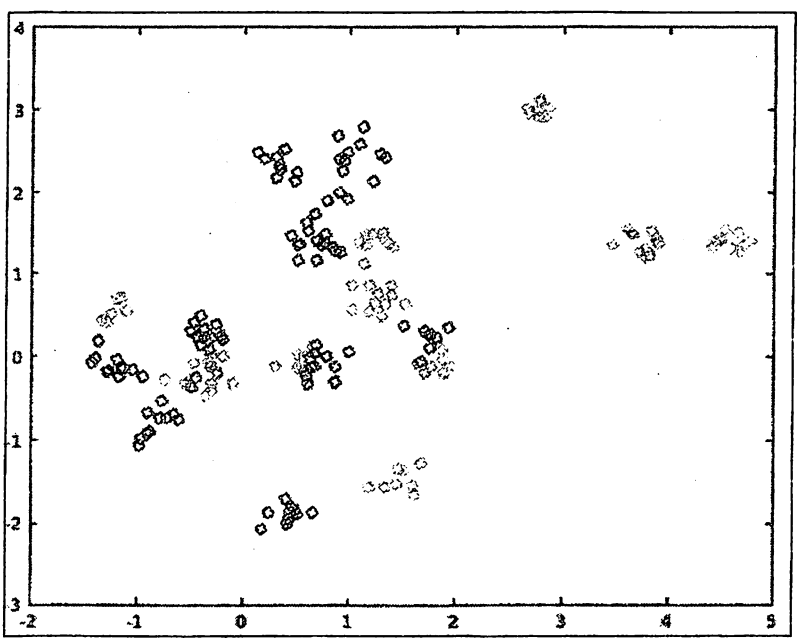


Рисунок 8.2 – 100 базисных точек

Подготовка данных для классификации

Поместите данные в одну матрицу и используйте функции векторного `grp` это помечает класс каждой точки, один указывает, что зеленый класс, и один указывает на красный класс.

```
cdata = [grnpts;redpts];
grp = ones(200,1);
grp(101:200) = -1;
```

Подготовка перекрестной проверки

Настройте раздел для перекрестной проверки.

```
c = cvpartition(200,'KFold',10);
```

Этот шаг является дополнительным. Если вы задаете раздел для оптимизации, то можно вычислить фактическую потерю перекрестной проверки для возвращенной модели.

Оптимизация подгонки

Найти хорошую подгонку, означая один оптимальными гиперпараметрами, которые минимизируют потерю перекрестной проверки, Байесовую оптимизацию использования. Задайте список гиперпараметров, чтобы оптимизировать при помощи OptimizeHyperparameters аргумент значения имени, и задает опции оптимизации при помощи HyperparameterOptimizationOptions аргумент значения имени.

Задайте 'OptimizeHyperparameters' как 'auto'. 'auto' опция включает типичный набор гиперпараметров, чтобы оптимизировать. fitcsvm находит оптимальные значения BoxConstraint и KernelScale. Установите опции гипероптимизации параметров управления использовать раздел перекрестной проверки c и выбрать 'expected-improvement-plus' функция захвата для воспроизводимости. Функция захвата по умолчанию зависит от времени выполнения и, поэтому, может дать различные результаты.

```
opts =
struct('CVPartition',c,'AcquisitionFunctionName','expected-
improvement-plus');
Mdl = fitcsvm(cdata,grp,'KernelFunction','rbf', ...
'OptimizeHyperparameters','auto','HyperparameterOptimization
Options',opts)
```

Iter	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	BoxConstraint	KernelScale
1	Best	0.345	0.22192	0.345	0.345	0.08474	306.44
2	Best	0.115	0.16686	0.115	0.12678	430.31	1.4864
3	Accept	0.52	0.14285	0.115	0.1152	0.028415	0.014369
4	Accept	0.61	0.14104	0.115	0.11564	133.94	0.0031427
5	Accept	0.34	0.14998	0.115	0.11504	0.018993	5.7742
6	Best	0.085	0.14041	0.085	0.085039	885.63	0.68483
7	Accept	0.105	0.13736	0.085	0.085428	0.3057	0.58118
8	Accept	0.21	0.13978	0.085	0.09566	0.16044	0.91824
9	Accept	0.085	0.16771	0.085	0.09725	972.19	0.45259
10	Accept	0.1	0.15169	0.085	0.090952	990.29	0.491
11	Best	0.08	0.13653	0.08	0.079362	2.5195	0.291
12	Accept	0.09	0.12055	0.08	0.08482	14.338	0.44386
13	Accept	0.1	0.12305	0.08	0.08508	0.0022577	0.23803
14	Accept	0.11	0.12852	0.08	0.087378	0.2115	0.32109
15	Best	0.07	0.1381	0.07	0.081507	910.2	0.25218
16	Best	0.065	0.1715	0.065	0.072457	953.22	0.26253
17	Accept	0.075	0.18371	0.065	0.072554	998.74	0.23087
18	Accept	0.295	0.14336	0.065	0.072647	996.18	44.626
19	Accept	0.07	0.15007	0.065	0.08946	985.37	0.27389
20	Accept	0.165	0.13721	0.065	0.071622	0.065103	0.13679

Iter	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	BoxConstraint	KernelScale
21	Accept	0.345	0.12409	0.065	0.071764	971.7	999.01
22	Accept	0.61	0.12674	0.065	0.071967	0.0010168	0.0010005
23	Accept	0.345	0.12977	0.065	0.071959	0.0011459	995.89
24	Accept	0.35	0.12457	0.065	0.071863	0.0010003	48.628
25	Accept	0.24	0.18461	0.065	0.072124	996.55	10.423
26	Accept	0.61	0.14087	0.065	0.072067	994.71	0.0010063
27	Accept	0.47	0.13224	0.065	0.07218	993.69	0.029723
28	Accept	0.3	0.12408	0.065	0.072291	993.15	170.81
29	Accept	0.16	0.27101	0.065	0.072103	992.81	3.8594
30	Accept	0.365	0.12737	0.065	0.072112	0.0010017	0.044287

- Objective — Значение целевой функции вычисляется в новом наборе гиперпараметров.
- Objective runtime — Время оценки целевой функции.
- Eval result — Отчет результата в виде Accept, Best, или Error. Accept указывает, что целевая функция возвращает конечное значение и Error указывает, что целевая функция возвращает значение, которое не является конечным действительным скаляром. Best указывает, что целевая функция возвращает конечное значение, которое ниже, чем ранее вычисленные значения целевой функции.
- BestSoFar(observed) — Минимальное значение целевой функции вычисляется до сих пор. Это значение является любой значением целевой функции текущей итерации (если Eval result значением для текущей итерации является Best) или значение предыдущего Best итерация.
- BestSoFar(estim.) — В каждой итерации программное обеспечение оценивает верхние доверительные границы значений целевой функции, с

помощью обновленной Гауссовой модели процесса, во всех наборах гиперпараметров, которые попробовали до сих пор. Затем программное обеспечение выбирает точку с минимальной верхней доверительной границей. BestSoFar(estim.) значение является значением целевой функции, возвращенным predictObjective функция в минимальной точке.

Пример выполнения практической работы № 8

Обучение классификатора SVM при помощи гауссова ядра.

В этом примере показано, как сгенерировать нелинейный классификатор с Гауссовой функцией ядра. Во-первых, сгенерируйте один класс точек в единичном диске в двух измерениях и другой класс точек в кольце от радиуса 1 к радиусу 2. Затем генерирует классификатор на основе данных с Гауссовым радиальным ядром основной функции. Линейный классификатор по умолчанию является очевидно неподходящим для этой проблемы, поскольку модель циркулярная симметричный. Установите параметр ограничения поля на Inf сделать строгую классификацию, не означая неправильно классифицированных учебных точек. Другие функции ядра не могут работать с этим строгим ограничением поля, поскольку они могут не мочь обеспечить строгую классификацию. Даже при том, что rbf классификатор может разделить классы, результат может быть перетренирован.

Сгенерируйте 100 точек, равномерно распределенных на единичном диске. Для этого сгенерируйте радиус r как квадратный корень из универсальной случайной переменной, сгенерируйте угол t однородно в $(0, 2\pi)$, и помещенный точка в $(r\cos(t), r\sin(t))$.

```
rng(1); % For reproducibility
r = sqrt(rand(100,1)); % Radius
t = 2*pi*rand(100,1);  % Angle
data1 = [r.*cos(t), r.*sin(t)]; % Points
```

Сгенерируйте 100 точек, равномерно распределенных в кольце. Радиус снова пропорционален квадратному корню, на этот раз квадратный корень из равномерного распределения от 1 до 4.

```
r2 = sqrt(3*rand(100,1)+1); % Radius
t2 = 2*pi*rand(100,1);      % Angle
data2 = [r2.*cos(t2), r2.*sin(t2)]; % points
```

Постройте точки и постройте круги радиусом 1 и 2 для сравнения.

```
figure;
plot(data1(:,1),data1(:,2),'r.','MarkerSize',15)
```

```
hold on
plot(data2(:,1),data2(:,2),'b.','MarkerSize',15)
ezpolar(@(x)1);ezpolar(@(x)2);
axis equal
hold off
```

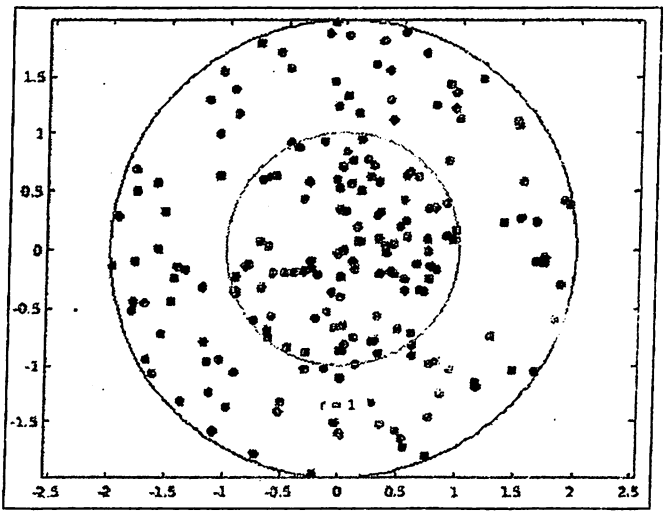


Рисунок 8.3 – Данные классификатора SVM в виде точек, сосредоточенные в кругах радиусом 1 и 2

Поместите данные в одну матрицу и сделайте вектор из классификаций.

```
data3 = [data1;data2];
theclass = ones(200,1);
theclass(1:100) = -1;
```

Обучите классификатор SVM с `KernelFunction` установите на `'rbf'` и `BoxConstraint` установите на `Inf`. Постройте контур решения и отметьте векторы поддержки.

```
%Train the SVM Classifier
cl = fitcsvm(data3,theclass,'KernelFunction','rbf',...
    'BoxConstraint',Inf,'ClassNames',[-1,1]);

% Predict scores over the grid
d = 0.02;
[x1Grid,x2Grid] =
meshgrid(min(data3(:,1)):d:max(data3(:,1)),...
```

```
    min(data3(:,2)):d:max(data3(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
[~,scores] = predict(cl,xGrid);

% Plot the data and the decision boundary
figure;
h(1:2) = gscatter(data3(:,1),data3(:,2),theclass,'rb','.');
hold on
ezpolar(@(x)1);
h(3) =
plot(data3(cl.IsSupportVector,1),data3(cl.IsSupportVector,2)
,'ko');
contour(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid)),[0
0],'k');
legend(h,{'-1','+1','Support Vectors'});
axis equal
hold off
```

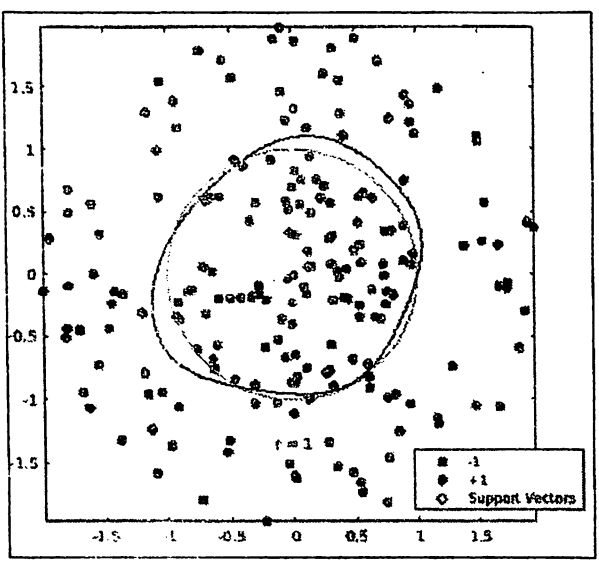


Рисунок 8.4 - Контур решения и вектора поддержки.

Обучение классификатора SVM, используя пользовательское ядро

В этом примере показано, как использовать пользовательскую функцию ядра, такую как сигмоидальное ядро, чтобы обучить классификаторы SVM и настроить пользовательские параметры функции ядра. Сгенерируйте

случайный набор точек в модульном кругу. Пометьте точки в первых и третьих квадрантах как принадлежащий положительному классу и тем во вторых и четвертых квадрантах в отрицательном классе.

```
rng(1);  % For reproducibility
n = 100; % Number of points per quadrant
r1 = sqrt(rand(2*n,1));                    % Random radii
t1 = [pi/2*rand(n,1); (pi/2*rand(n,1)+pi)]; % Random angles
for Q1 and Q3
X1 = [r1.*cos(t1) r1.*sin(t1)];            % Polar-to-
Cartesian conversion
r2 = sqrt(rand(2*n,1));
t2 = [pi/2*rand(n,1)+pi/2; (pi/2*rand(n,1)-pi/2)]; % Random
angles for Q2 and Q4
X2 = [r2.*cos(t2) r2.*sin(t2)];
X = [X1; X2];          % Predictors
Y = ones(4*n,1);
Y(2*n + 1:end) = -1; % Labels
```

Отобразите данные на графике.

```
figure;
gscatter(X(:,1),X(:,2),Y);
title('Scatter Diagram of Simulated Data')
```

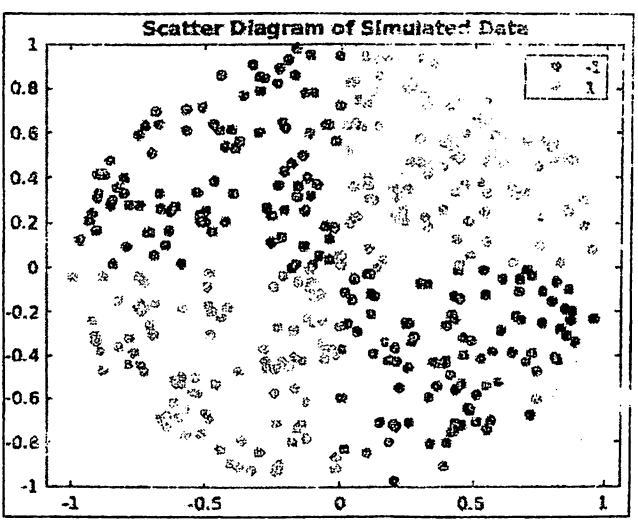


Рисунок 8.5 - Данные классификатора SVM в виде точек

Запишите функцию, которая принимает две матрицы в пространстве признаков как входные параметры и преобразовывает их в матрицу Грамма использование сигмоидального ядра.

```
function G = mysigmoid(U,V)
% Sigmoid kernel function with slope gamma and intercept c
gamma = 1;
c = -1;
G = tanh(gamma*U*V' + c);
end
```

Сохраните этот код как файл с именем mysigmoid на вашем пути MATLAB®.

Обучите классификатор SVM с помощью сигмоидальной функции ядра, это хороший способ стандартизации данных.

```
Mdl1 = fitcsvm(X,Y,'KernelFunction','mysigmoid','Standardize',true);
```

Mdl1 ClassificationSVM классификатор, содержащий предполагаемые параметры.

Отобразите данные на графике и идентифицируйте векторы поддержки и контур решения.

```
 % Compute the scores over a grid
d = 0.02; % Step size of the grid
[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...
    min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];        % The grid
[~,scores1] = predict(Mdl1,xGrid); % The scores
figure;
h(1:2) = gscatter(X(:,1),X(:,2),Y);
hold on
h(3) = plot(X(Mdl1.IsSupportVector,1),...
    X(Mdl1.IsSupportVector,2),'ko','MarkerSize',10);
    % Support vectors
contour(x1Grid,x2Grid,reshape(scores1(:,2),size(x1Grid)),[0
0],'k');
    % Decision boundary
title('Scatter Diagram with the Decision Boundary')
legend({'-1','1','Support Vectors'},'Location','Best');
hold off
```

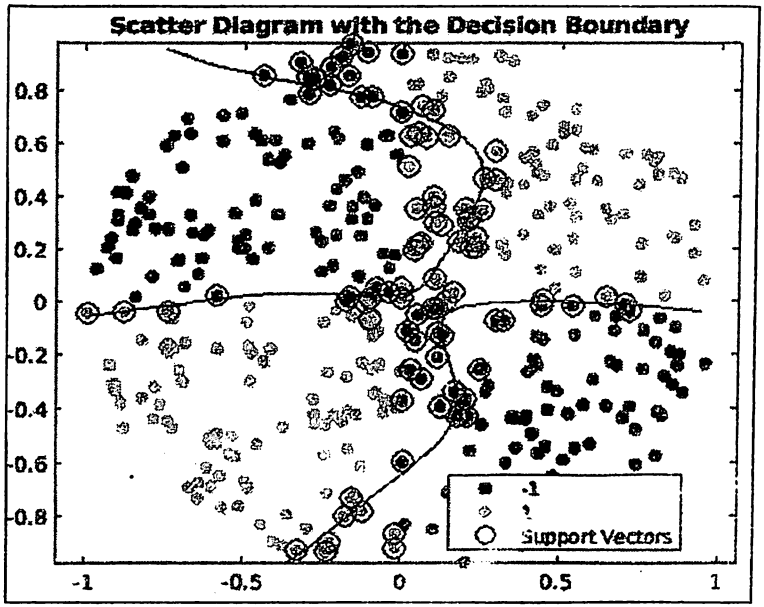


Рисунок 8.6 - Данные на графике с векторами поддержки и контуром решения.

Можно настроить параметры ядра в попытке улучшить форму контура решения. Эта сила также уменьшает misclassification уровень в выборке.

Определите misclassification уровень из выборки при помощи 10-кратной перекрестной проверки.

```
CVMdl1 = crossval(Mdl1);
misclass1 = kfoldLoss(CVMdl1);
misclass1
misclass1 =
    0.1350
misclassification уровень из выборки составляет 13,5%.
```

Задание для выполнения практической работы № 8

Задание 1. Обучите классификатор SVM при помощи гауссова ядра. Выведите на экран точечное изображение данных заключенных в кругах, график с векторами поддержки и контурами решения

Задание 2. Обучите классификатор SVM, используя пользовательское ядро. Выведите на экран данны... классификатора SVM в виде точек и график с векторами поддержки и контуром решения

Содержание и оформление отчета:

1. Наличие титульного листа, шрифт Times New Roman – 14, интервал 1,5;
2. Выполнение всех приложенных заданий;
3. Заключение по выполненной работе;
4. Ответы на контрольные вопросы.

Контрольные вопросы

1. Дайте описание методу опорных векторов
2. Дайте описание классификатору SVM
3. Опишите синтаксис классификатора SVM
4. Где используется классификатор SVM?
5. Для чего используется классификатор SVM?

Практическая работа № 9
КЛАССИФИЦИРОВАНИЕ ДАННЫХ В СРЕДЕ MATLAB С ПОМОЩЬЮ МЕТОДА kNN

Цель работы: Изучение метода kNN, классифицирование данных с помощью метода kNN в среде MATLAB.

kNN расшифровывается как *k Nearest Neighbor* или *k Ближайших Соседей* — это один из самых простых алгоритмов классификации, также иногда используемый в задачах регрессии. Благодаря своей простоте, он является хорошим примером, с которого можно начать знакомство с областью Machine Learning. В данной статье рассмотрен пример написания кода такого классификатора на python, а также визуализация полученных результатов.

Задача классификации в машинном обучении — это задача отнесения объекта к одному из заранее определенных классов на основании его формализованных признаков. Каждый из объектов в этой задаче представляется в виде вектора в N-мерном пространстве, каждое измерение в котором представляет собой описание одного из признаков объекта. Допустим нам нужно классифицировать мониторы: измерениями в нашем пространстве параметров будут величина диагонали в дюймах, соотношение сторон, максимальное разрешение, наличие HDMI-интерфейса, стоимость и др.

Алгоритм

Для классификации каждого из объектов тестовой выборки необходимо последовательно выполнить следующие операции:

- Вычислить расстояние до каждого из объектов обучающей выборки
- Отобрать k объектов обучающей выборки, расстояние до которых минимально
- Класс классифицируемого объекта — это класс, наиболее часто встречающийся среди k ближайших соседей

Учитывая набор X точек n и функции расстояния, k - самый близкий сосед (k NN) поиск позволяет вам найти k самые близкие точки в X к точке запроса или набору точек Y. Метод поиска NNk и k основанные на NN алгоритмы широко используются в качестве правил изучения сравнительного теста. Относительная простота метода поиска NN k дает возможность сравнивать результаты других методов классификации с k результаты NN. Метод использовался в различных областях, таких как:

- биоинформатика
- обработка изображений и сжатие данных

- поиск документов
- компьютерное зрение
- мультимедийная база данных
- маркетинг анализа данных

Можно использовать k поиск NN, для поиска других алгоритмов машинного обучения, такие как:

- k классификация NN
- локальная взвешенная регрессия
- недостающее обвинение данных и интерполяция
- оценка плотности

Можно также использовать k поиск NN со многими основанными на расстоянии функциями изучения, такими как кластеризация K-средних значений.

В отличие от этого для положительного вещественного значения r, rangesearch находит все точки в X это - на расстоянии r из каждой точки в Y. Этот поиск фиксированного радиуса тесно связан с k поиск NN, когда это поддерживает те же метрики расстояния и поисковые классы, и использует те же алгоритмы поиска.

k- поиск NN используя исчерпывающий поиск

Когда ваши входные данные соответствуют любому из следующих критериев, knnsearch использует метод исчерпывающего поиска по умолчанию, чтобы найти k - самые близкие соседи:

- Количество столбцов X больше чем 10.
- X issparse .
- Метрика расстояния также:
  - 'seuclidean'
  - 'mahalanobis'
  - 'cosine'
  - 'correlation'
  - 'spearman'
  - 'hamming'
  - 'jaccard'
  - Пользовательская функция расстояния

knnsearch также использует метод исчерпывающего поиска, если вашим поисковым объектом является ExhaustiveSearcher объект модели. Метод исчерпывающего поиска находит расстояние от каждой точки запроса до каждой точки в X, оценивает их в порядке возрастания и возвращает точки k с

наименьшими расстояниями. Например, эта схема показывает $k = 3$ самых близких соседа.

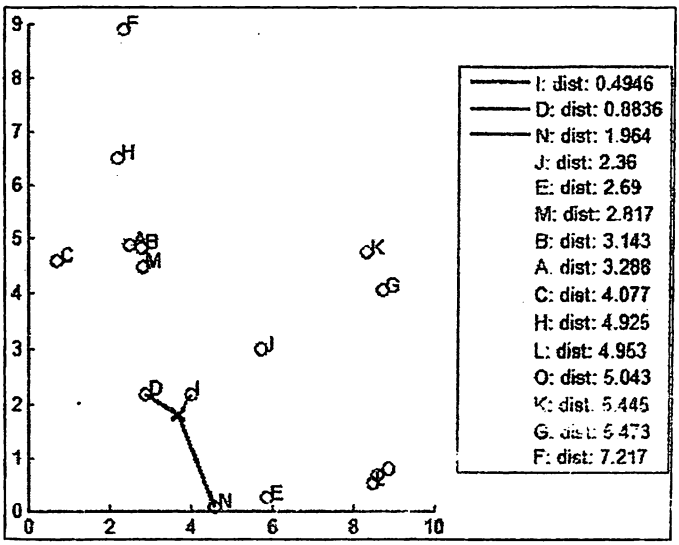


Рисунок 9.1 - *k*- поиск NN используя исчерпывающий поиск

k*- поиск ближайшего соседа, используя d-дерево *K

Когда ваши входные данные соответствуют всем следующим критериям, `knnsearch` создает d-дерево *K* по умолчанию, чтобы найти *k* - самые близкие соседи:

- Количество столбцов X меньше 10.
- X не разреженно.
- Метрика расстояния также:
  - `'euclidean'` (значение по умолчанию)
  - `'cityblock'`
  - `'minkowski'`
  - `'chebychev'`

`knnsearch` также использует d-дерево *K*, если вашим поисковым объектом является `KDTreeSearcher` объект модели.

D-деревья *K* делят ваши данные на узлы с в большей части `BucketSize` (значение по умолчанию равняется 50), точки на узел, на основе координат (в противоположность категориям). Следующие схемы иллюстрируют эту концепцию с помощью `patch` объекты к цветовому коду различные "блоки".

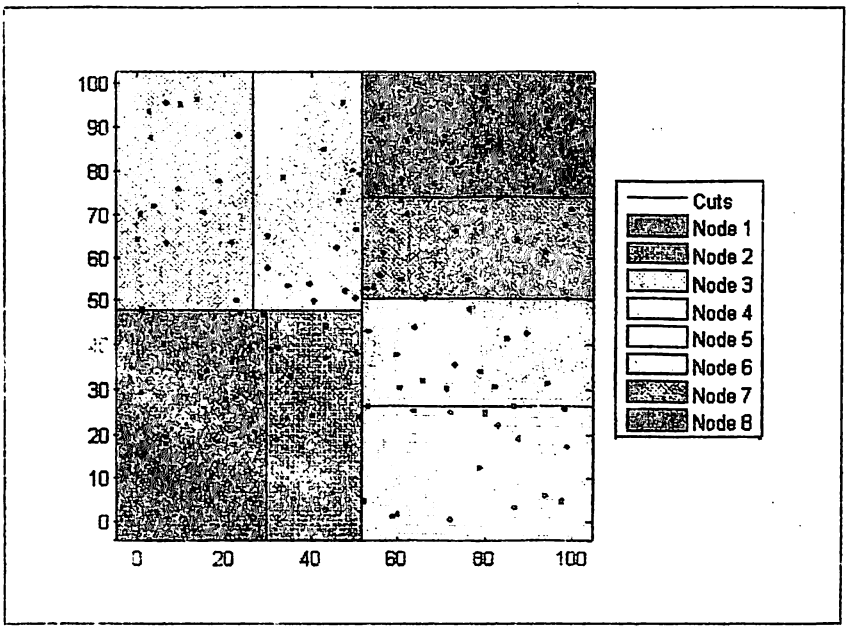


Рисунок 9.2 – Разделение данных на блоки

Когда это необходимо, найти k–поиск NN данной точки запроса, knnsearch делает следующее:

1. Определяет узел, которому принадлежит точка запроса. В следующем примере точка запроса (32,90) принадлежит Узлу 4.
2. Находит самые близкие точки k в том узле и его расстоянии до точки запроса. В следующем примере точки в красных кругах являются равноотстоящими от точки запроса и являются самыми близкими точками к точке запроса в Узле 4.
3. Выбирает все другие узлы, имеющие любую область, которая является на том же расстоянии, в любом направлении, от точки запроса до k th самая близкая точка. В этом примере, только Узел 3 перекрытия чистый черный круг, сосредоточенный в точке запроса с радиусом, равняются расстоянию до самых близких точек в Узле 4.
4. Узлы поисковых запросов в той области значений для любых точек ближе к точке запроса. В следующем примере точка в красном квадрате немного ближе к точке запроса, чем те в Узле 4.

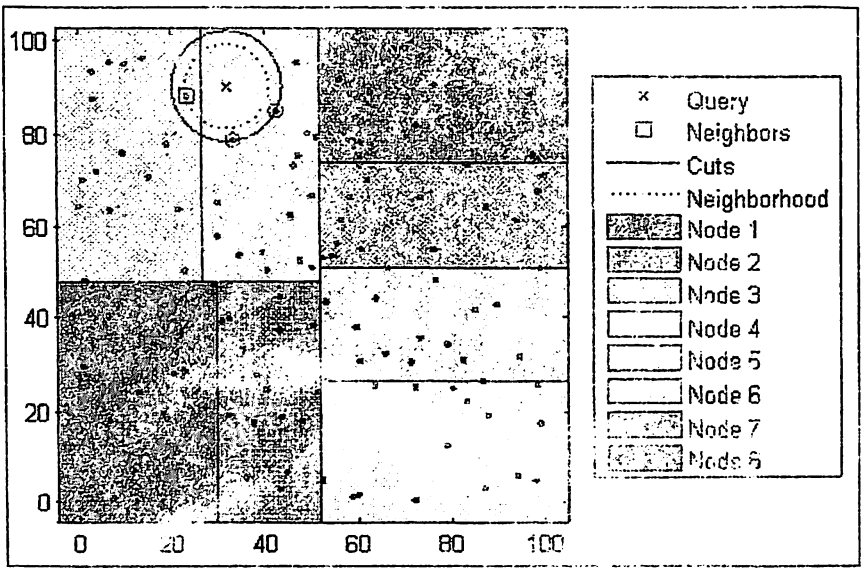


Рисунок 9.3 - *k*- поиск ближайшего соседа, используя d-дерево *K*

Используя d-дерево *K* для больших наборов данных меньше чем с 10 размерностями (столбцы) может быть намного более эффективным, чем использование метода исчерпывающего поиска, как knnsearch потребности вычислить только подмножество расстояний. Чтобы максимизировать КПД d-деревьев *K*, используйте KDTreeSearcher модель.

Пример классификации данных:

```
% Первый набор данных
>> mu1=[0,0]; %среднее
>> S1=[.1 0;0 .1]; % ковариации
% Генерация данных распределения Гаусса
>> data1=mvnrnd(mu1, S1, 100);
% Второй набор данных
>> mu2=[1.25 1.25];
>> S2=[.1 0;0 .1];
>> data2=mvnrnd(mu2,S2,100);
% Третий набор данных
>> mu3=[-1.25;1.25];
>> S3=[.1 0;0 .1];
>> data3=mvnrnd(mu3,S3,100);
% Вывод данных на экран
>> plot(data1(:,1),data1(:, 2),'b+');
```

```
>> hold off
>> plot(data1(:,1),data1(:, 2),'b+');
>> hold on;
>> plot(data2(:,1),data2(:,2),'r+');
>> plot(data3(:,1),data3(:,2),'g+');
```

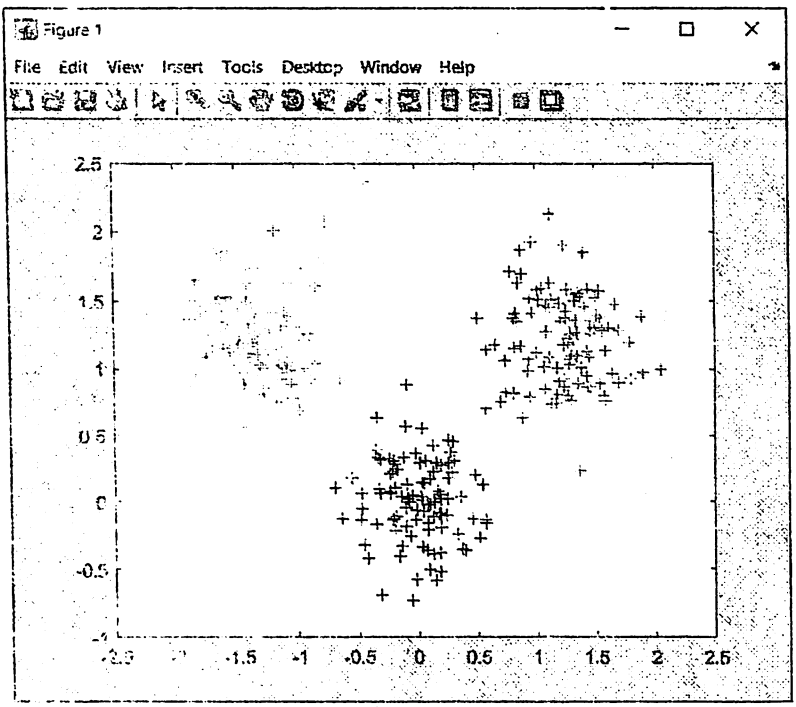


Рисунок 9.4 - Классификация введенных данных

Пример выполнения практической работы № 9

Классификация данных о запросе.

Классифицируйте новую точку на основе последних двух столбцов данных Фишера. Только использование последних двух столбцов облегчает построение.

```
load fisheriris
x = meas(:,3:4);
gscatter(x(:,1),x(:,2),species)
legend('Location','best')
```

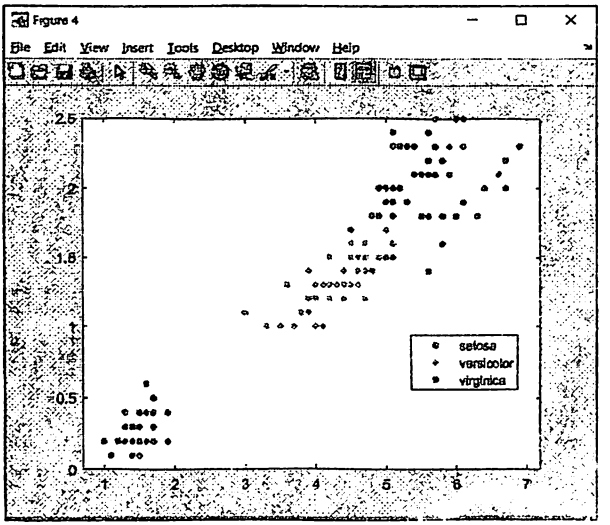


Рисунок 9.5 – Построение графика по сгенерированным данным

Постройте новую точку.

```
newpoint = [5 1.45];
line(newpoint(1),newpoint(2),'marker','x','color','k',...
   'markersize',10,'linewidth',2)
```

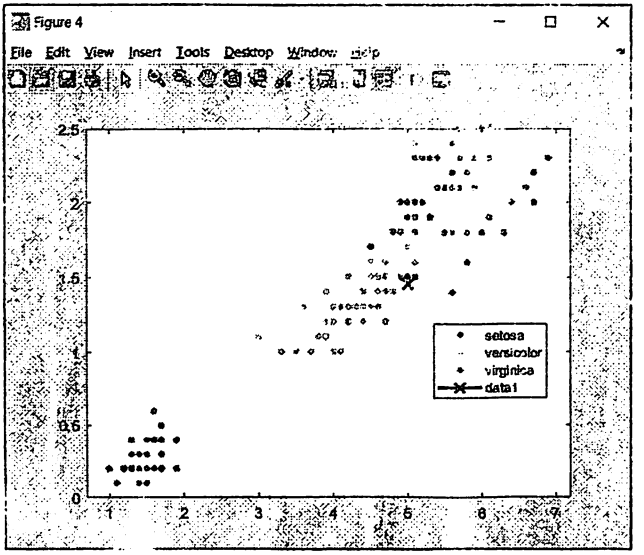


Рисунок 9.6 - Построение новой точки

Kd-дерево граничат с моделью искателя.

```
Mdl = KDTreeSearcher(x)
Mdl =
  KDTreeSearcher with properties:

       BucketSize: 50
         Distance: 'euclidean'
    DistParameter: []
                X: [150x2 double]
```

Mdl KDTreeSearcher модель. По умолчанию метрика расстояния, которую используют, чтобы искать соседей, является Евклидовым расстоянием.

Найдите 10 точек выборки самыми близкими к новой точке.

```
[n,d] = knnsearch(Mdl,newpoint,'k',10);
line(x(n,1),x(n,2),'color',[.5 .5 .5],'marker','o',...
    'linestyle','none','markersize',10)
```

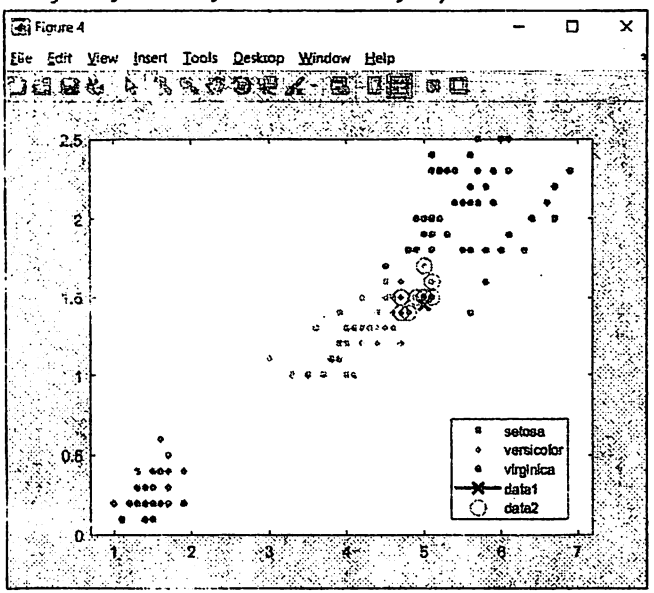


Рисунок 9.7 – 10 ближайших точек к новой точке

Это результат поиска с помощью knnsearch, который нашел только самые близкие восемь соседних точек. На самом деле этот конкретный набор данных содержит дублирующиеся значения.

```
x(n,:)
ans = 10×2

    5.0000    1.5000
    4.9000    1.5000
    4.9000    1.5000
    5.1000    1.5000
    5.1000    1.6000
    4.8000    1.4000
    5.0000    1.7000
    4.7000    1.4000
    4.7000    1.4000
    4.7000    1.5000
```

Сделайте оси равными, таким образом, расчетные расстояния соответствуют очевидным расстояниям на равной оси графика и увеличение, чтобы видеть соседей лучше.

```
xlim([4.5 5.5]);
ylim([1 2]);
axis square
```

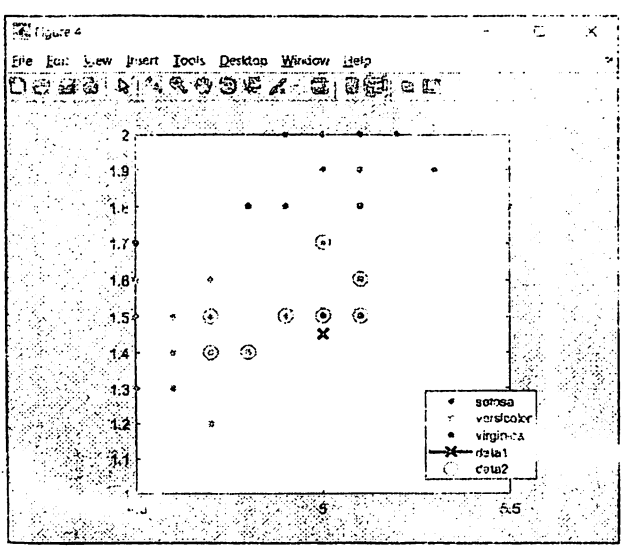


Рисунок 9.8 - Изменение параметров оси x и y, для того что бы видеть наглядное расстояние между точками.

Найдите разновидности 10 соседей.

```
tabulate(species(n))
       Value     Count    Percent
   virginica         2     20.00%
  versicolor         8     80.00%
```

Используя правило на основе решения большинством голосов 10 самых близких соседей, можно классифицировать эту новую точку как versicolor.

Визуально идентифицируйте соседей путем рисования круга вокруг группы их. Задайте центр и диаметр круга, на основе местоположения новой точки.

```
ctr = newpoint - d(end);
diameter = 2*d(end);
% Draw a circle around the 10 nearest neighbors.
h = rectangle('position',[ctr,diameter,diameter],...
   'curvature',[1 1]);
h.LineStyle = ':';
```

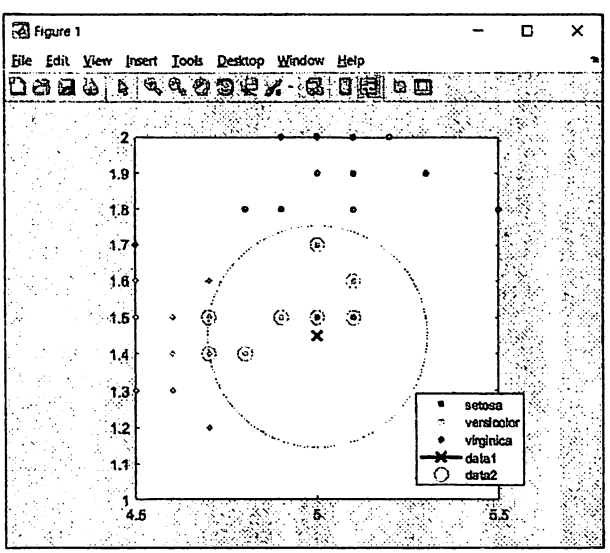


Рисунок 9.9 - Визуальная идентификация ближайших точек

Используя тот же набор данных, найдите 10 самых близких соседей трех новых точек.

```
figure
newpoint2 = [5 1.45;6 2;2.75 .75];
gscatter(x(:,1),x(:,2),species)
legend('location','best')
[n2,d2] = knnsearch(Mdl,newpoint2,'k',10);
line(x(n2,1),x(n2,2),'color',[.5 .5 .5],'marker','o',...
   'linestyle','none','markersize',10)
line(newpoint2(:,1),newpoint2(:,2),'marker','x','color','k',
...
   'markersize',10,'linewidth',2,'linestyle','none')
```

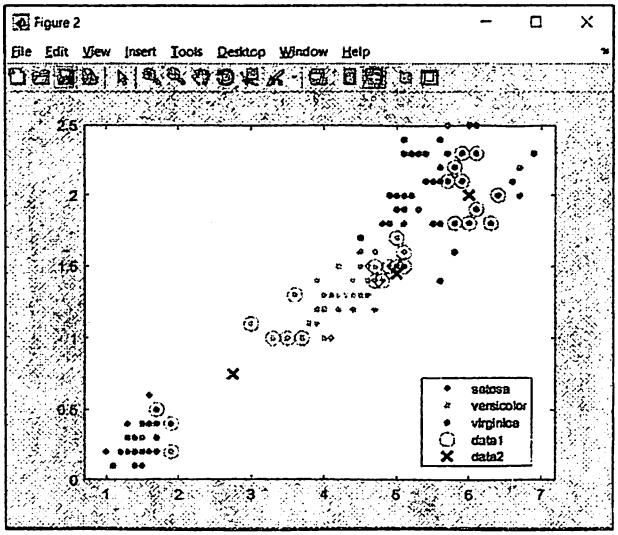


Рисунок 9.10 - Нахождение 10 ближайших точек, к 3 новым точкам

Найдите разновидности 10 самых близких соседей к каждой новой точке.

```
tabulate(species(n2(1,:)))
       Value     Count   Percent
   virginica         2     20.00%
  versicolor         8     80.00%
tabulate(species(n2(2,:)))
      Value     Count   Percent
  virginica        10    100.00%
tabulate(species(n2(3,:)))
```

```
      Value      Count    Percent
 versicolor          7     70.00%
     setosa          3     30.00%
```

Нахождение близких соседей, используя пользовательскую метрику расстояния

В этом примере показано, как найти индексы трех самых близких наблюдений в X к каждому наблюдению в Y относительно расстояния хи-квадрата. Эта метрика расстояния используется в анализе соответствия, ссобенно в экологических приложениях.

Случайным образом сгенерируйте нормально распределенные данные в две матрицы. Количество строк может варьироваться, но количество столбцов должно быть равным. Этот пример использует 2D данные для графического вывода.

```
rng(1) % For reproducibility
X = randn(50,2);
Y = randn(4,2);
h = zeros(3,1);
figure
h(1) = plot(X(:,1),X(:,2),'bx');
hold on
h(2) = plot(Y(:,1),Y(:,2),'rs','MarkerSize',10);
title('Heterogeneous Data')
```

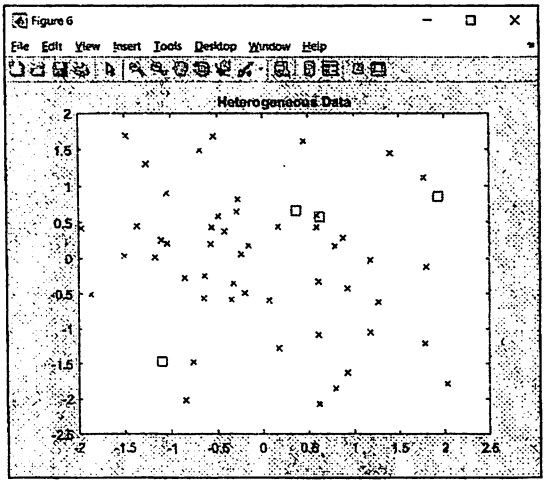


Рисунок 9.11 - Вывод сгенерированных данных

Строки X и Y соответствуйте наблюдениям, и столбцы являются, в целом, размерностями (например, предикторы).

Расстояние хи-квадрата между *j-dimensional точками x и z*

$$\chi(x,z) = \sqrt{\sum_{j=1}^{J} w_j (x_j - z_j)^2},$$

где w_j вес, сопоставленный с размерностью j.

Выберите веса для каждой размерности и задайте функцию расстояния хи-квадрата. Функция расстояния должна:

- Возьмите в качестве входных параметров одну строку X, например, x, и матричный Z.
- Сравните x к каждой строке Z.
- Возвратите векторный D из длины n_Z, где n_Z количество строк Z. Каждый элемент D расстояние между наблюдением, соответствующим x и наблюдения, соответствующие каждой строке Z.

```
w = [0.4; 0.6];
chiSqrDist = @(x,Z)sqrt((bsxfun(@minus,x,Z).^2)*w);
```

Этот пример использует произвольные веса для рисунка.

Найдите индексы трех самых близких соседей в X к каждому соседу в Y.

```
k = 3;
[Idx,D] = knnsearch(X,Y,'Distance',chiSqrDist,'k',k);
idx и D 4 3 матрицы.
```

- idx(j,1) индекс строки самого близкого наблюдения в X к наблюдению j Y, и D(j,1) их расстояние.
- idx(j,2) индекс строки следующего ближайшего наблюдения в X к наблюдению j Y, и D(j,2) их расстояние.
- И так далее.

Идентифицируйте самые близкие точки на графике.

```
for j = 1:k
    h(3)
plot(X(Idx(:,j),1),X(Idx(:,j),2),'ko','MarkerSize',10);
end
legend(h,{'\texttt{X}','\texttt{Y}','Nearest
Neighbor'},'Interpreter','latex')
title('Heterogeneous Data and Nearest Neighbors')
hold off
```

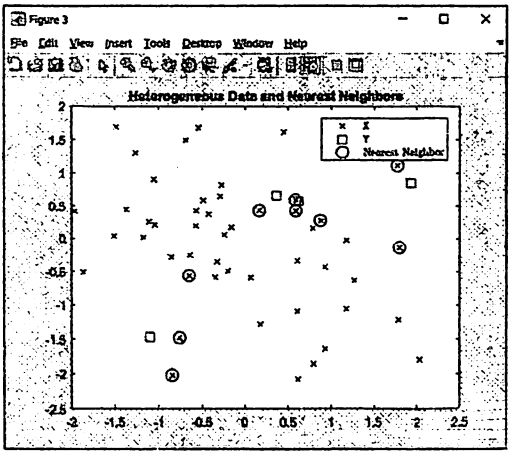


Рисунок 9.12 - Ближайшие идентифицированные данные

Задание для выполнения практической работы № 9

Задание 1. Сгенерировать матрицу данных. Построить график сгенерированных данных, в виде точек.

Задание 2. Задать новую точку. С помощью функции «knnsearch» найти ближайшие точки и выделить их.

Задание 3. Далее, задать две новые точки, и выделить ближайшие к ним точки.

Задание 4. Вывести ближайшие соседние точки, используя пользовательскую метрику расстояния

Содержание и оформление отчета:

1. Наличие титульного листа, шрифт Times New Roman – 14, интервал 1,5;
2. Выполнение всех приложенных заданий;
3. Заключение по выполненной работе;
4. Ответы на контрольные вопросы.

Контрольные вопросы

1. Для чего нужна классификация данных?
2. Опишите методы классификации данных.
3. В чем заключаются задачи классификации?
4. Опишите метод исчерпывающего поиска
5. Опишите поиск ближайшего соседа с помощью d – дерева.

Практическая работа № 10

КЛАСТЕРИЗАЦИЯ ДАННЫХ В СРЕДЕ MATLAB. K-MEANS КЛАСТЕРИЗАЦИИ

Цель работы: Изучение кластеризации данных. Выполнить K-means кластеризацию

Метод *k*-средних (англ. *k-means*) — наиболее популярный метод кластеризации. Был изобретён в 1950-х годах математиком Гуго Штейнгаузом и почти одновременно Стюартом Ллойдом. Особую популярность приобрёл после работы Маккуина.

Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

$$V = \sum_{i=1}^{k} \sum_{x \in S_i} (x - \mu_i)^2$$

где k — число кластеров, S_i полученные кластеры, i=1,2,...,k, а μ_i — центры масс всех векторов S_i из кластера .

По аналогии с методом главных компонент центры кластеров называются также *главными точками*, а сам метод называется *методом главных точек*[4] и включается в общую теорию *главных объектов*, обеспечивающих наилучшую аппроксимацию данных.

Алгоритм

Алгоритм представляет собой версию EM-алгоритма, применяемого также для разделения смеси гауссиан. Он разбивает множество элементов векторного пространства на заранее известное число кластеров k.

Основная идея заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.

Алгоритм завершается, когда на какой-то итерации не происходит изменения внутрикластерного расстояния. Это происходит за конечное число итераций, так как количество возможных разбиений конечного множества конечно, а на каждом шаге суммарное квадратичное отклонение V уменьшается, поэтому зацикливание невозможно.

Как показали Дэвид Артур и Сергей Васильвицкий, на некоторых классах множеств сложность алгоритма по времени, нужному для сходимости, равна $2^{\Omega(\sqrt{n})}$.

Демонстрация алгоритма

Действие алгоритма в двумерном случае. Начальные точки выбраны случайно.

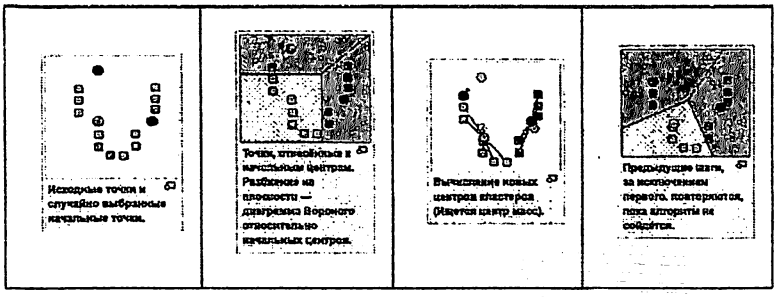


Рисунок 10.1 - Действие алгоритма в двумерном случае

Проблемы k-means:

- Не гарантируется достижение глобального минимума суммарного квадратичного отклонения V, а только одного из локальных минимумов.
- Результат зависит от выбора исходных центров кластеров, их оптимальный выбор неизвестен.
- Число кластеров надо знать заранее.

Применение для задач глубокого обучения и машинного зрения

В алгоритмах глубокого обучения метод k-средних иногда применяют не по прямому назначению (классификация разбивкой на кластеры), а для создания так называемых фильтров (ядер свёртки, словарей).

Например, для распознавания изображений в алгоритм k-средних подают небольшие случайные кусочки изображений обучающей выборки, допустим, размером 16x16 в виде линейного вектора, каждый элемент которого кодирует яркость своей точки. Количество кластеров k задается большим, например 256. Обученный метод k-средних при определенных условиях вырабатывает при этом центры кластеров (центроиды), которые представляют собой удобные базисы, на которые можно разложить любое входное изображение. Такие "обученные" центроиды в дальнейшем используют в качестве фильтров, например для свёрточной нейронной сети в качестве ядер свёртки или других аналогичных систем машинного зрения. Таким образом осуществляется обучение без учителя при помощи метода k-средних.

Использование алгоритма кластеризации K-средних для сегментации изображений

Рисунок 10.2 - Оригинальное изображение, для импорта в Matlab

На результат кластеризации k-средних влияет количество K выбранных центров кластеров и их начальные положения, а также геометрические свойства и порядок считывания образцов шаблона. В практических приложениях необходимо протестировать разные значения K и выбрать разные начальные значения центров кластеров.

Причина в следующем, но мы используем функцию k-means в MATLAB напрямую, и нам не нужно рассматривать начальную точку кластеризации. В коде мы сосредоточены на K. Лучше всего поместить эффекты разных K на один и тот же рисунок, чтобы эффект был более очевидным в сравнении. Кроме того, на результаты с большей вероятностью повлияют геометрические свойства образцов узора, то есть соотношение переднего и заднего плана изображения, распределение RGB и т. Д. (Поэтому рекомендуется попробовать еще несколько картинок, чтобы увидеть сравнение)

Пример выполнения практической работы № 10

Пример импорта и кластеризации изображения.

```
close all;
clear all;
clc;
k=2;
org = imread ('C:\1480439597175676615.jpg');%
прочитайте изображение
```

```
figure;
subplot(2,2,1);
imshow (org), title ('Оригинальное изображение');%
Показать исходное изображение
```

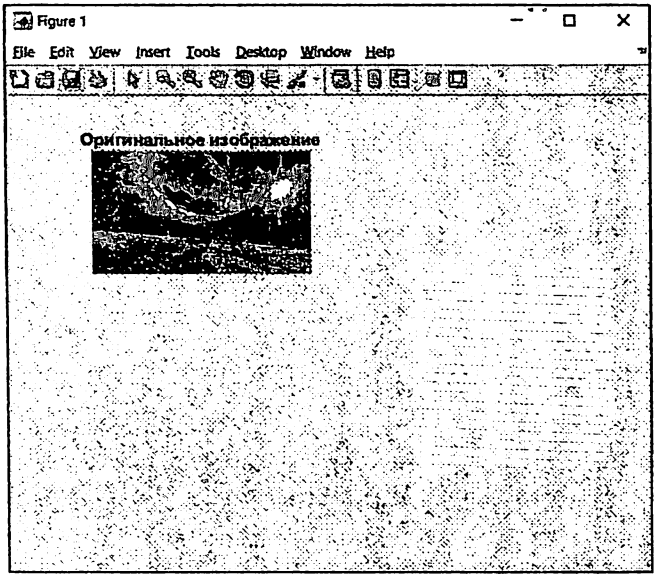


Рисунок 10.3 – Вывод первого, оригинального изображения

```
 % Далее вам необходимо знать размер изображения (длину
и ширину), если вы работаете непосредственно с
изображением RGB, как показано ниже.
 % [m, n, p] = size (org)% m, n - требование, p = 3 -
количество каналов
  % Или используйте следующий метод для преобразования
в оттенки серого, а затем найдите
gray=rgb2gray(org);
[m,n]=size(gray);
  % RGB-3-канальная декомпозиция изображения
 % org (:,:, 1) ... представляют канал rgb
A = reshape(org(:, :, 1), m*n, 1);
B = reshape(org(:, :, 2), m*n, 1);
C = reshape(org(:, :, 3), m*n, 1);
 data = [A B C];% r g b формируют характеристики
выборки, каждая выборка имеет три значения атрибута,
всего выборки ширины * высоты
```

```
 % Второе изображение
 res = kmeans (double (data), k);% вызывает встроенную
функцию kmeans
 result = reshape (res, m, n);% обратное преобразование
в форму изображения
subplot(2,2,2);
 % label2rgb предназначена для преобразования матрицы
меток в изображение RGB
 imshow (label2rgb (result)), title (strcat ('K = ',
num2str (k), ' результат сегментации канала RGB '));%
отобразить результат сегментации
```

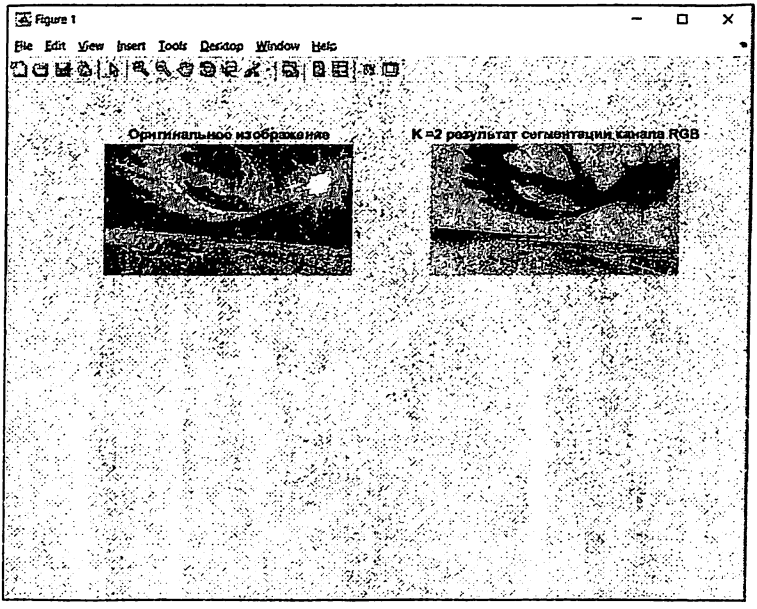


Рисунок 10.4 – Вывод второго изображения, с K = 2 сегментацией каналов RGB

```
 % Третья картинка
res = kmeans(double(data), k+1);
result = reshape(res, m, n);
subplot(2,2,3);
 imshow (label2rgb (result)), title (strcat ('K = ',
num2str (k + 1), ' результат сегментации канала RGB'));
```

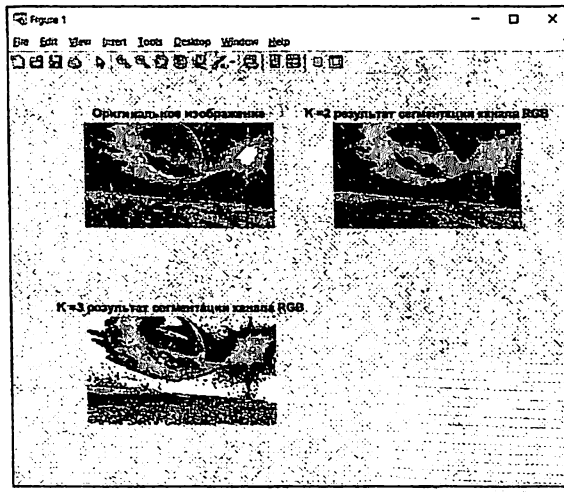


Рисунок 10.5 – Вывод второго изображения, с K = 2 сегментацией каналов RGB

```
 % Четвертое изображение
res = kmeans(double(data), k+2);
result = reshape(res, m, n);
subplot(2,2,4);
 imshow (label2rgb (result)), title (strcat ('K = ',
num2str (k + 2), ' результат сегментации канала RGB'));
```

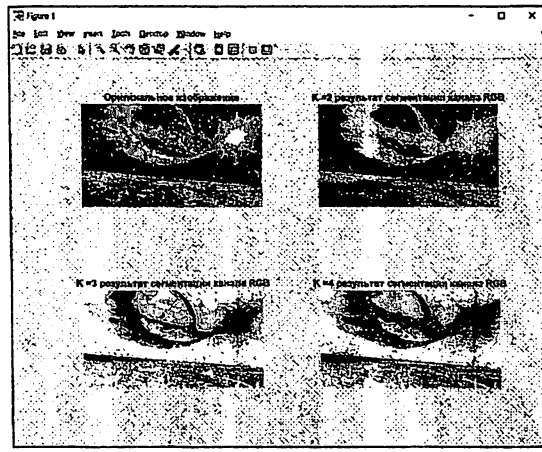


Рисунок 10.6 – Вывод второго изображения, с K = 2 сегментацией каналов RGB

Пример кластеризации данных

```
>> %% генерация данных
>> K = 3;
>> numObservarations = 100;
>> dimensions = 3;
>> data = rand([numObservarations dimensions]);
>> %% кластер
>> opts = statset('MaxIter', 500, 'Display', 'iter');
>> [clustIDX, clusters, interClustSum, Dist] = kmeans(data,
K, 'options',opts, ...
'distance','sqEuclidean',          'EmptyAction','singleton',
'replicates',3);

  iter     phase         num           sum
     1         1         100       14.8781
     2         1           5       14.4825
     3         1           2       14.4253
     4         1           2       14.3799
     5         1           2       14.2439
     6         1           2       14.1372
     7         1           2       14.0336
     1         1         100       15.1086
     2         1           8       14.3616
     3         1           4       14.1798
     4         1           3       14.0821
     5         1           1       14.0391
     6         1           3       13.9901
     1         1         100       14.4034
     2         1           5       14.0544
     3         1           1       14.0322
     4         1           1       13.9951
     5         1           2       13.9541
     6         1           1       13.9252
Best total sum of distances = 13.9252
>> %% построение данных + кластеров
>> figure, hold on
>>   scatter3(data(:,1),data(:,2),data(:,3),   50,   clustIDX,
'filled')
```

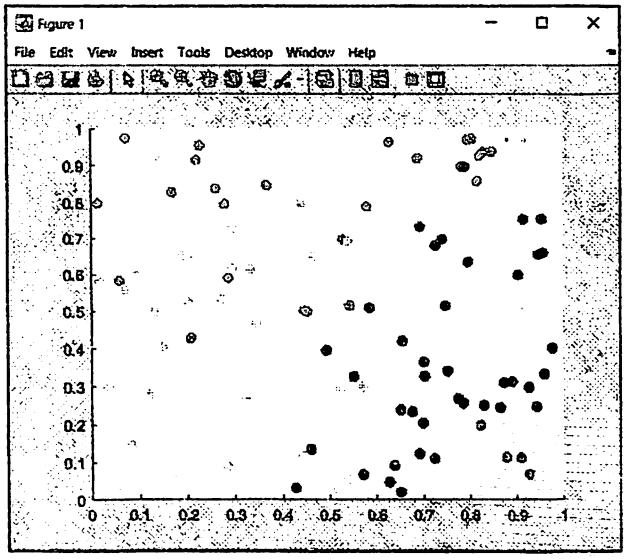


Рисунок 10.7 – Изображение данных

```
>> scatter3(clusters(:,1),clusters(:,2),clusters(:,3), 200,
(1:K)', 'filled')
>> hold off, xlabel('x'), ylabel('y'), zlabel('z')
```

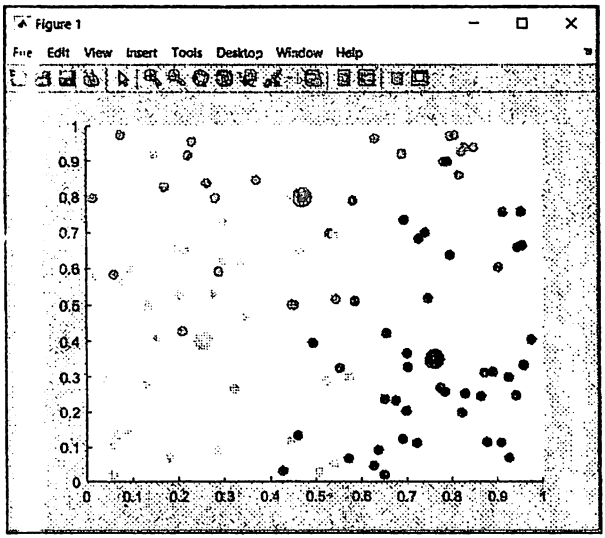


Рисунок 10.8 – Кластерные данные и кластерные цетроиды

```
>> %% построение качества кластеров
>> figure
>> [silh,h] = silhouette(data, clustIDX);
>> avrgScore = mean(silh);
```

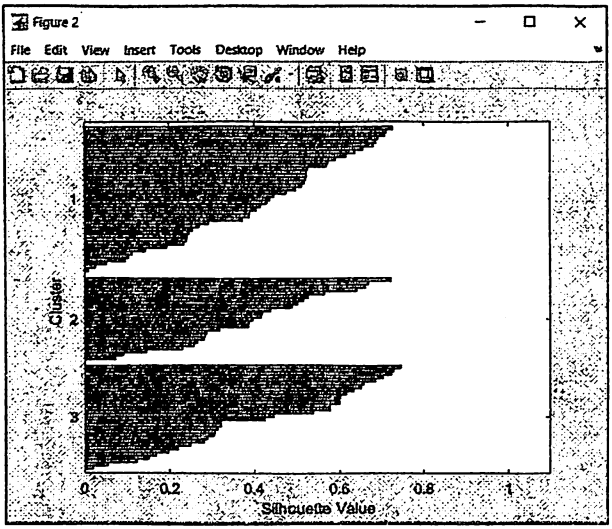


Рисунок 10.9 – Построение качества кластеров

```
>> %% Назначьте данные кластерам
>> % вычислить расстояние (в квадрате) всех экземпляров до
каждого центроида кластера
>> D = zeros(numObservarations, K); % расстояния инициализации
>> for k=1:K
%d = sum((x-y).^2).^0.5
D(:,k)    =    sum(    ((data    -    repmat(clusters(k,:),
numObservarations,1)).^2), 2);
end
>> % поиск для всех экземпляров кластерного шкафа
>> [minDists, clusterIndices] = min(D, [], 2);
>> sum(clusterIndices == clustIDX)

ans =

   100
```

Задание для выполнение практической работы № 10

Задание 1. Импортировать изображение в Matlab. Провести сегментацию каналов RGB загруженного изображения с различными значениями K.

Задание 2. Провести кластеризацию данных. Вывести графики данных, кластеров и кластерных цетроидов.

Содержание и оформление отчета:

1. Наличие титульного листа, шрифт Times New Roman – 14, интервал 1,5;
2. Выполнение всех приложенных заданий;
3. Заключение по выполненной работе;
4. Ответы на контрольные вопросы.

Контрольные вопросы

1. Как работает кластеризация?
2. Для чего нужен кластерный анализ?
3. Для чего используется кластеризация?
4. Опишите основную идею алгоритма k-means
5. Что влияет на результат кластеризации k-means?
6. Где используется алгоритм k-means?

Практическая работа № 11
АНАЛИЗ ДАННЫХ НА ОСНОВЕ ПАКЕТА FLT В СИСТЕМЕ MATLAB.

Цель работы: Изучение пакета FLT. Анализ данных на основе пакета FLT в среде Matlab.

Свойства зрительной системы человека

Fuzzy Logic Toolbox (FLT) – это пакет расширения MATLAB, содержащий инструменты для проектирования систем нечеткой логики.

Пакет позволят создавать экспертные системы на основе нечеткой логики, проводить кластеризацию нечеткими алгоритмами, а также проектировать нечеткие нейросети.

Глаз человека является уникальным механизмом, обеспечивающим адаптивную настройку в соответствии с внешними условиями.

Рассмотрим некоторые основные свойства зрительной системы человека. Важной характеристикой зрительной системы является чувствительность, т.е. способность реагировать на внешние изменения. Чувствительность характеризуется верхним и нижним абсолютными порогами.

Существует несколько различных видов чувствительности. Световая чувствительность характеризует свойство глаза реагировать на максимально малый световой поток. Однако здесь следует отметить, что вероятность распознавания максимально малого светового потока зависит также и от других факторов, например угла зрения.

Зрительная система по-разному реагирует на излучения, которые равны по мощности, но излучаемые из различных диапазонов спектра. Такая чувствительность называется спектральной.

Способность глаза различать минимальные различия яркости смежных областей изображения характеризуется контрастной чувствительностью. Также зрительная система характеризуется различной чувствительностью к цветовому тону, т.е. к излучениям из различных участков спектра. Зрительная система характеризуется еще чувствительностью к насыщенности цвета.

Приведенные выше типы чувствительности зрительной системы не являются постоянными, а зависят от многих факторов, в частности, условий освещения. Например, при переходе из темной комнаты в светлую, нужно некоторое время для восстановления светочувствительности глаза. Этот процесс называется яркостной адаптацией глаза.

Цветоощущение характеризуется тремя основными характеристиками – светлота, цветовой тон и насыщенность. Для классификации цветов используются цветовые пространства.

На основе свойств и характеристик зрительных систем создаются различные модели цветового зрения. Среди них следует выделить модель цветового зрения, предложенную Фреем. Особенностью этой модели является то, что зрительная система представлена тремя каналами, два из которых характеризуют цветность, а третий – яркость. Эта модель наиболее удачно согласуется со многими свойствами цветного зрения.

Пример выполнения практической работы № 11

Возможности цифровой обработки изображений в Matlab

На сегодняшний день среда Matlab, в частности пакет прикладных программ Image Processing Toolbox, является наиболее мощным инструментом для моделирования и исследования методов обработки изображений. Он включает большое количество встроенных функций, реализующих наиболее распространенные методы обработки изображений. Рассмотрим основные возможности пакета Image Processing Toolbox.

Геометрические преобразования изображений

К наиболее распространенным функциям геометрических преобразований относится кадрирование изображений (imcrop), изменение размеров (imresize) и поворот изображения (imrotate).

```
>> L=imread('cameraman.tif');
>> imshow(L);
```

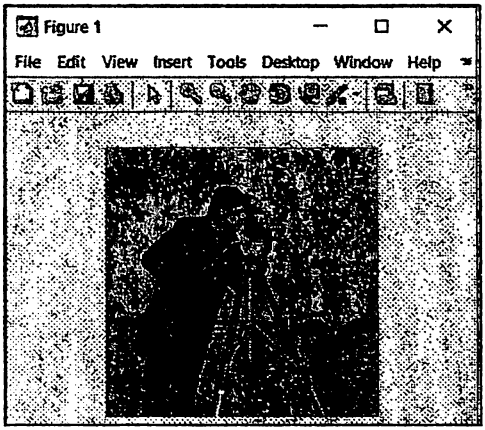


Рисунок 11.1 – Загрузка изображения в Matlab.

```
>> imcrop;
```

Суть кадрирования состоит в том, что функция imcrop позволяет с помощью мыши в интерактивном режиме вырезать часть изображения и поместить ее в новое окно просмотра.

Рисунок 11.2 - Кадрирование изображения с помощью мыши

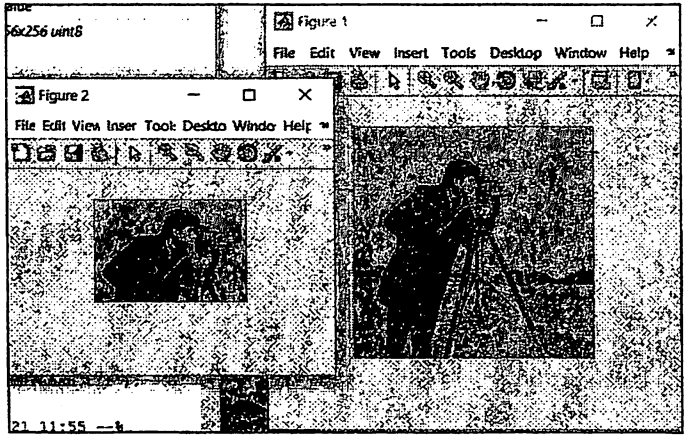


Рисунок 11.3 - Результат кадрирования изображения с помощью imcrop.

Функция изменения размеров изображения imresize позволяет, используя специальные методы интерполяции, изменять размер любого типа изображения.

В пакете Image Processing Toolbox существует функция imrotate, которая осуществляет поворот изображения на заданный угол.

```
L1=imrotate(L,35,'bicubic');
figure,imshow(L1)
```

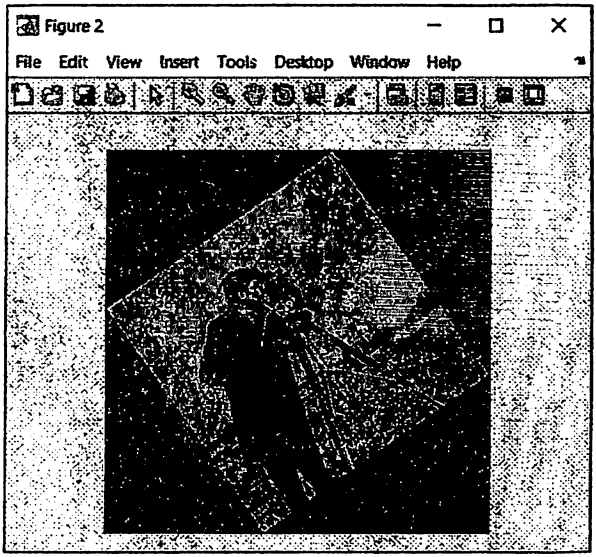


Рисунок 11.4 - Поворот изображения на заданный угол

Таким образом, приведенные выше функции позволяют поворачивать, вырезать части, масштабировать, т.е. работать с целым массивом изображения.

Анализ изображений

Для работы с отдельными элементами изображений используются такие функции как imhist, impixel, mean2, corr2 и другие.

Одной из наиболее важных характеристик является гистограмма распределения значений интенсивностей пикселей изображения, которую можно построить с помощью функции imhist.

```
L=imread('cameraman.tif');
figure, imshow(L);
figure, imhist(L);
```

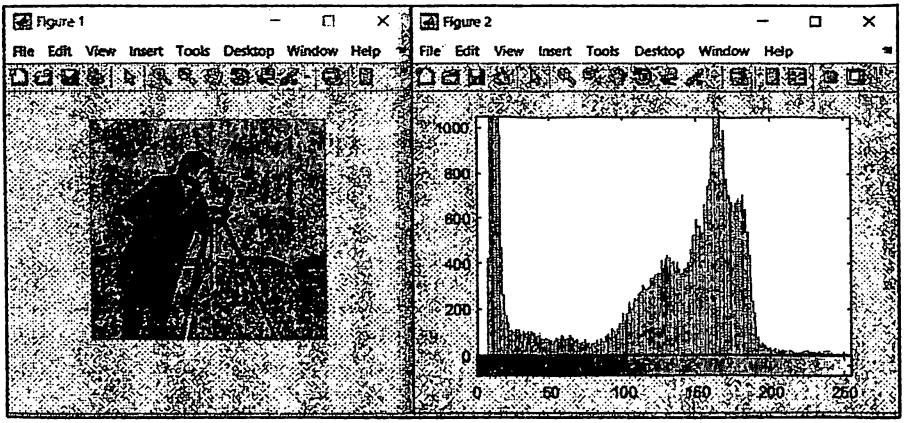


Рисунок 11.5 - Гистограмма распределения значений интенсивностей пикселей изображения

Довольно часто при проведении анализа изображений возникает необходимость определить значения интенсивностей некоторых пикселей. Для этого в интерактивном режиме можно использовать функцию impixel.

```
>> impixel
```

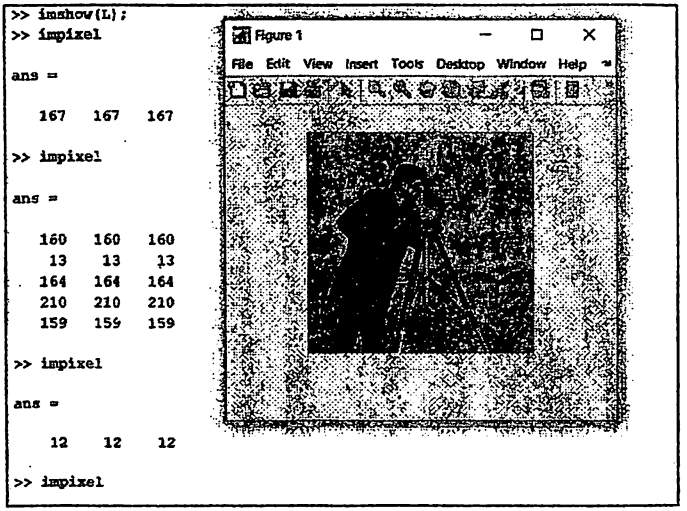


Рисунок 11.6 - Работа функции impixel

Следует отметить, что функция impixel по своим возможностям в некоторой степени повторяет опцию Data Cursor, пример использования которой приведен на изображении внизу.

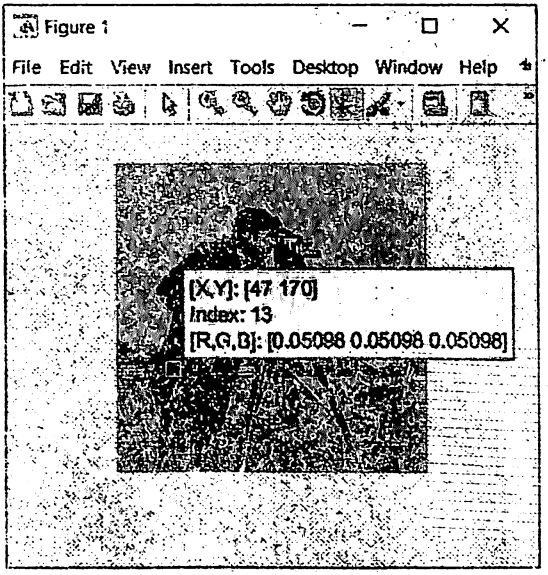


Рисунок 11.7 - Вывод пиксельных значений

Еще одной широко применяемой функцией является функция mean2 – она вычисляет среднее значение элементов матрицы.

Функция corr2 вычисляет коэффициент корреляции между двумя матрицами. Другими словами, с помощью функции corr2 можно сказать насколько две матрицы или изображения похожи между собой. Эта функция широко применяется при решении задач распознавания.

Улучшение изображений

Среди встроенных функций, которые реализуют наиболее известные методы улучшения изображений, выделим следующие – histeq, imadjust та imfilter(fspecial).

Как уже отмечалось ранее, гистограмма изображения является одной из наиболее информативных характеристик. На основе анализа гистограммы можно судить о яркостных искажениях изображения, т.е. сказать о том, является ли изображение затемненным или засветленным. Известно, что в идеале на цифровом изображении в равном количестве должны присутствовать пиксели со всеми значениями яркостей, т.е. гистограмма

должна быть равномерной. Перераспределение яркостей пикселей на изображении с целью получения равномерной гистограммы выполняет метод эквализации, который в среде Matlab реализован в виде функции histeq.

```
>> L=imread('cameraman.tif');
>> figure, imshow(L);
>> L1=histeq(L);
>> Figure, imsow(L1);
```

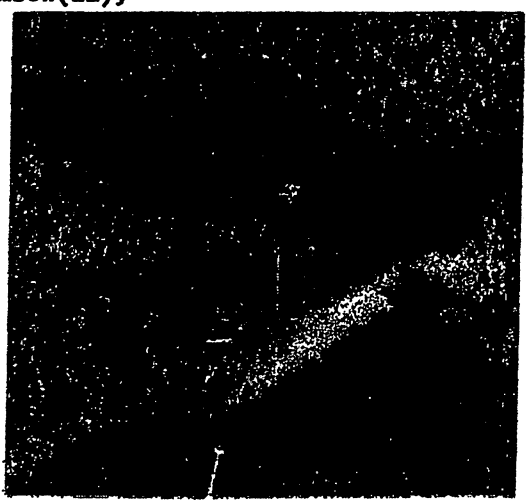


Рисунок 11.8 - Исходное изображение

Рисунок 11.9 - Изображение после эквализации гистограммы

Довольно часто при формировании изображений не используется весь диапазон значений интенсивностей, что отрицательно отражается на качестве визуальных данных. Для коррекции динамического диапазона сформированных изображений используется функция imadjust.

```
>> L=imread('cameraman.tif');
>> figure, imshow(L);
>> L1=imadjust(L);
>> figure, imshow(L1);
>> figure, imhist(L);
>> figure, imhist(L1);
```

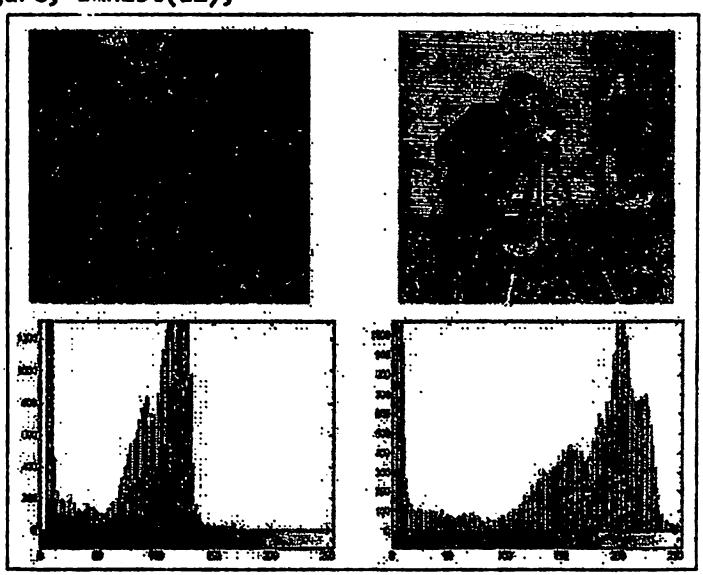


Рисунок 11.10 - Коррекции динамического диапазона

Также при решении задач улучшения изображений используется функция imfilter в паре с функцией fspecial. Функция fspecial позволяет задавать различные типы масок фильтра. Рассмотрим пример использования маски фильтра, повышающего резкость изображения.

```
>> L=imread('cameraman.tif');
>> figure, imshow(L);
>> H = fspecial('unsharp');
>> L1 = imfilter(L,H,'replicate');
>> figure, imshow(L1);
```

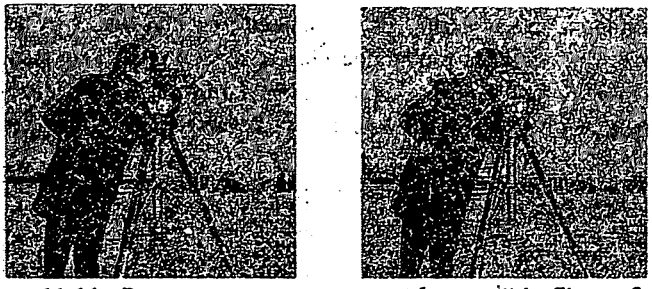


Рисунок 11.11 - Результат использования функций imfilter и fspecial

Фильтрация изображений

Пакет Image Processing Toolbox обладает очень мощным инструментарием по фильтрации изображений. Среди множества встроенных функций, которые решают задачи фильтрации изображений, особо следует выделить fspecial, ordfilt2, medfilt2.

Функция fspecial является функцией задания маски предопределенного фильтра. Эта функция позволяет формировать маски:

- высокочастотного фильтра Лапласа;
- фильтра, аналогичного последовательному применению фильтров Гаусса и Лапласа, так называемого лапласиана-гауссиана;
- усредняющего низкочастотного фильтра;
- фильтра, повышающего резкость изображения.

Рассмотрим примеры применения названных выше фильтров:

```
>> L=imread('cameraman.tif');
>> figure, imshow(L);
>> h=fspecial('laplasian',.5);
>> L1 = imfilter(L,h,'replicate');
>> figure, imshow(L1);
```

Рисунок 11.12 - высокочастотного фильтра Лапласа;

```
>> h=fspecial('log', 3, .5);
>> L1 = imfilter(L,h,'replicate');
>> figure, imshow(L1);
```

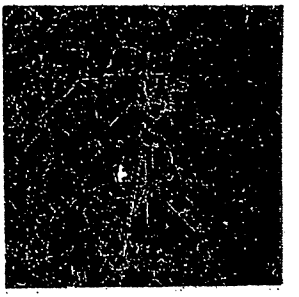


Рисунок 11.13 - Применение лапласиана-гауссиана

```
>> h=fspecial('average', 3);
>> L1 = imfilter(L,h,'replicate');
>> figure, imshow(L1);
```

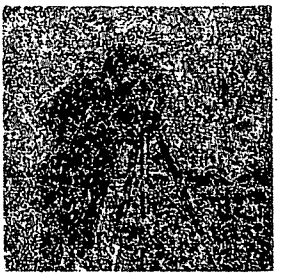


Рисунок 11.14 - Применение усредняющего низкочастотного фильтра

```
>> h=fspecial('unsharp', .5);
>> L1 = imfilter(L,h,'replicate');
>> figure, imshow(L1);
```

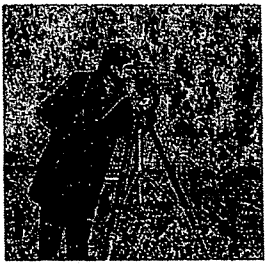


Рисунок 11.15 – Применение фильтра, повышающего резкость изображения

Сегментация изображений

Среди встроенных функций пакета Image Processing Toolbox, которые применяются при решении задач сегментации изображений, следует выделить qtdecomp, edge и roicolor.

Функция qtdecomp выполняет сегментацию изображения методом разделения и анализа однородности не перекрывающихся блоков изображения.

```
>> I = imread('cameraman.tif');
>> S = qtdecomp(I,.27);
>> blocks = repmat(uint8(0),size(S));
>> for dim = [512 256 128 64 32 16 8 4 2 1];
  numblocks = length(find(S==dim));
  if (numblocks > 0)
    values = repmat(uint8(1),[dim dim numblocks]);
    values(2:dim,2:dim,:) = 0;
       blocks = qtsetblk(blocks,S,dim,values);
  end
end

>> blocks(end,1:end) = 1;
>> blocks(1:end,end) = 1;

>> imshow(I), figure, imshow(blocks,[])
```

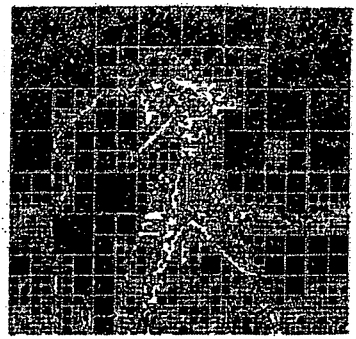


Рисунок 11.16 - Сегментация изображения

Одной из наиболее часто применяемых является функция выделения границ edge, которая реализует такие встроенные методы – Собела, Превит, Робертса, лапласиан-гауссиана, Канни и др.

Приведем примеры реализации функции edge с использованием различных фильтров.

```
>> clear;
>> I = imread('cameraman.tif');
>> BW1=edge(I,'sobel');
>> figure,imshow(BW1);title('sobel');
>> BW2=edge(I,'prewitt');
>> figure,imshow(BW2);title('prewitt');
>> BW3=edge(I,'roberts');
>> figure,imshow(BW3);title('roberts');
>> BW4=edge(I,'log');
>> figure,imshow(BW4);title('log');
>> BW5=edge(I,'zerocross');
>> figure,imshow(BW5);title('zerocross');
>> BW6=edge(I,'canny');
>> figure,imshow(BW6);title('canny');
```

Задание для выполнения практической работы № 11

Задание 1. Необходимо произвести цифровую обработку изображения на Matlab, используя функции описанные выше.

Содержание и оформление отчета:

1. Наличие титульного листа, шрифт Times New Roman – 14, интервал 1,5;
2. Выполнение всех приложенных заданий;
3. Заключение по выполненной работе;
4. Ответы на контрольные вопросы.

Контрольные вопросы

1. Что вы понимаете под обработкой данных?
2. Что вы понимаете под анализом данных?
3. Для чего нужен анализ данных?
4. Для чего нужен паке FLT (Fuzzy Logic Toolbox)?

ОГЛАВЛЕНИЕ

Бичими 60x84 1/16. Босма табоғи 7,5
Адади 10. Буюртма - № 150
Тошкент ахборот технологиялари университети
"Мухаррирлик нашр" бўлимида чоп этилди.
Тошкент ш, Амир Темур кўчаси, 108-уй