МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ имени МУХАММАДА АЛ-ХОРАЗМИ

ФАКУЛЬТЕТ ТЕЛЕВИЗИОННЫЕ ТЕХНОЛОГИИ

КАФЕДРА «АУДИОВИЗУАЛЬНЫЕ ТЕХНОЛОГИИ»

МЕТОДИЧЕСКОЕ ПОСОБИЕ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ПРЕДМЕТУ «ОБРАБОТКА ЦИФРОВОЙ ИНФОРМАЦИИ»

Ташкент 2019

Автор: Касымова Ш.Т

«Методическое пособие по выполнению лабораторных работ по предмету «Обработка цифровой информации» /ТУИТ. 82 с. Ташкент, 2019.

Методические пособия предназначены для проведения лабораторных работ в первом семестре 4-го курса.

В первом семестре четвертого курса студенты должны изучить методы обработки цифровой информации, которые проводятся в системе МАТЛАБ. После ознакомления с указанной системой студенты используя возможности **Image Processing Toolbox (IPT)** изучают работу с графиками в системе, простейшие операции с изображениями, пространственную фильтрацию изображений- преобразование яркости и контраста, подавление импульсных шумов, работу с файлами изображений.

В представляемом пособии каждая лабораторная работа снабжена теоретическим материалом и примерами выполнения, что поможет, самостоятельно усвоить прорабатываемый материал. Кроме того, в методическом пособии содержатся примерные темы заданий лабораторных работ.

Методическое пособие было рассмотрено на заседании учебно-методического Совета ТУИТ имени Мухаммада ал-Хорезми №4(116)от 23.10.2018 года, одобрено и рекомендовано к тиражированию.

© Ташкентский университет информационных технологий имени Мухаммада ал-Хоразми 2019 год

ЛАБОРАТОРНАЯ РАБОТА №1
ОСНОВЫ РАБОТЫ В MATLAB

1. ЦЕЛЬ РАБОТЫ

Изучение интерфейса системы MATLAB. Знакомство с основными объектами языка MATLAB, их вводом и визуализацией результатов вычислений. Получение практических навыков работы в командном режиме и в среде редактора m-файлов.

2. ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

MATLAB (от англ. MATrix LABoratory) – язык программирования высокого уровня и система инженерных и научных вычислений, разработчиком которого является корпорация MathWorks Inc. (США). В нем интегрированы вычисления, визуализация и программирование в удобной для пользователя среде. Типовой набор действий, которые могут производиться в MATLAB, включает:

– математические вычисления;

– разработку алгоритмов;

– моделирование и создание прототипных систем;

– анализ данных, их исследование и визуализацию;

– разработку приложений и пользовательского интерфейса.

MATLAB является интерактивной системой, особенностью которой является представление данных в виде массивов (матриц), а их преобразований в виде матричных операций. Использование такого подхода способствует значительному повышению скорости вычислений. Не смотря на то, что основы системы MATLAB были заложены в 80-х годах прошлого столетия, система находится в стадии постоянного развития и совершенствования. Это проявляется в создании множества наборов специализированных программ – **toolboxes** (наборы инструментов), которых насчитывается несколько десятков. Интеграция с программной системой **Simulink**, предназначенной для

MIDAGI

моделирования блочно заданных динамических систем и устройств, открывает новые возможности для разработчиков в среде визуально-ориентированного программирования. *Интерфейс* После запуска MATLAB на экране появляется основное окно системы, состоящее из 4 частей (рис. 1.1):

1) Command Window (Командное окно);

2) Workspace (Рабочее пространство);

3) Command History (История команд);

4) Current Folder (в некоторых версиях Current Directory – Текущая папка).

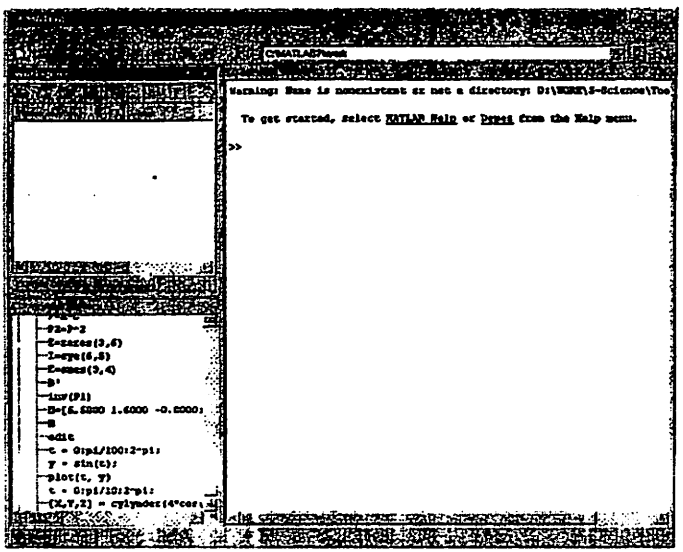

Рис. 1.1 – Структура основного окна системы MATLAB

Command Window (Командное окно) представляет собой окно, в котором осуществляется непосредственный ввод команд и отображается результат их выполнения. Окно *Workspace* (Рабочее пространство) содержит список текущих переменных и их размеры. Для редактирования переменных необходимо дважды щелкнуть мышью на имени необходимой переменной для отображения окна *Array Editor* (Редактор массива), в котором можно редактировать

отдельные элементы векторов и матриц. Окно *Command History* (История команд) отображает список ранее введенных команд в окне Command Window. *Current Folder* (Текущая папка) представляет собой окно, в котором выводится содержимое папки, имя которой отображается в раскрывающемся списке Current Folder на панели инструментов окна MATLAB. *Рабочее пространство Command Window* Посредством этого окна вводятся математические выражения, получаются результаты вычислений, а также выдаются сообщения, формируемые системой. *Режим прямых вычислений*, называемый также командным режимом представляет собой работу системы, при которой вычисления выполняются без составления программы. В режиме прямых вычислений сеанс работы до выхода из MATLAB называется *текущей сессией*. Объекты языка MATLAB вводятся построчно в командной строке в ходе текущей сессии. Входящие в сессию определения переменных и функций, расположенные в рабочей области памяти (но не саму сессию), можно записать на диск (файлы с расширением .mat), используя команду *save* (см. далее). В ходе текущей сессии при вводе команд в командную строку должны соблюдаться некоторые правила:

1) ввод осуществляется после символа «>>» в начале строки и завершается нажатием клавиши ENTER, при этом результаты вычислений выводятся в том же окне, а для рисунков открываются отдельные графические окна;

2) символ « ; » (точка с запятой) в конце строки блокирует автоматический вывод значений переменных;

3) символ « ... » (многоточие) в конце строки признак того, что следующая строка – продолжение предыдущей;

4) символ « % » означает, что текст после него – комментарий.

Встроенный редактор по мере ввода текста осуществляет синтаксический контроль. Результаты контроля отображаются цветовыми выделениями (табл. 1.1). цвета можно изменить на любые.

Таблица 1.1

Цветовые выделения системы MATLAB

Синтаксис	Цвет
Ключевые слова языка	Синий
Операторы, константы, переменные	Чёрный
Открытая строка	Фиолетовый
Закрытая строка	Пурпурный
Системные команды	Коричневый
Комментарий после знака « % »	Зелёный
Ошибки	Красный

Объекты языка MATLAB

Базовыми объектами языка MATLAB являются:

1) команды;

2) операторы;

3) константы;

4) переменные;

5) функции;

6) выражения.

Команда–объект языка MATLAB со стандартным именем, предназначенный для взаимодействия с системой. Команда имеет следующий формат:

<стандартное имя команды> <содержательная часть>,

где содержательная часть уточняется для конкретной команды, также может отсутствовать.

Рассмотрим несколько наиболее полезных команд:

1) Специальная команда ***help*** вызывает справку в системе MATLAB. Вызванная без параметров, она выдает оглавление пакетов функций, установленных в вашей системе. Если вы хотите узнать перечень функций конкретного пакета, скажем **ELFUN**, то нужно набрать:

help elfun.

Система выдаст полный список доступных элементарных функций. Чтобы просмотреть справку по назначению и синтаксису конкретной функции, скажем **SIN** нужно набрать:

help sin.

Справку по использованию системы MATLAB можно получить и через меню **HELP** основного окна или нажав клавишу **F1**.

2) Сохранение и восстановление переменных рабочей среды можно выполнить непосредственно из командной строки. Для этого служат команды ***save*** и ***load***. Для сохранения необходимо выполнить команду:

save FileName.

Если не указывать расширение, то MATLAB сохранит переменные рабочей среды в файле **FileName.mat**. В начале следующего сеанса работы для считывания переменных следует ввести команду:

load FileName.

Аналогичными являются действия с меню:

FILE --> SAVE WORKSPACE AS... и **FILE --> OPEN...**

3) Для очистки командного окна MATLAB используется команда ***clc***, которая оставляет неизменным содержимое буфера команд и рабочего пространства, то есть все переменные остаются сохраненными в памяти.

4) Для отображения имен всех переменных, размещенных в данный момент в рабочем пространстве, используется команда ***who***.

5) Получить сведения о конкретной переменной можно командой

whos имя1 имя2 ... .

6) Для стирания всех переменных в памяти используется команда *clear*. Для стирания некоторых переменных необходимо указывать их имена:

clear имя1 имя2 ... .

Оператор – это специальное обозначение для определенной операции над данными – операндами. Существует три категории операторов:

1) арифметические операторы используются для составления арифметических выражений и выполняют числовые вычисления (табл. 1.2);

2) операторы отношения используются для сравнения операндов (табл. 1.3);

3) логические операторы позволяют строить логические выражения (табл. 1.4).

Таблица 1.2

Арифметические операторы

Функция	Название	Оператор
plus	Плюс	+
unplus	Унарный плюс	+
minus	Минус	–
unminus	Унарный минус	–
mtimes	Матричное умножение	*
times	Поэлементное умножение массивов	.*
mpower	Возведение матрицы в степень	^
power	Поэлементное возведение массива в степень	.^
mldivide	Обратное (справа налево) деление матриц	\
mrdivide	Деление матриц справа налево	/
idivide	Поэлементное деление массивов справа налево	.\
rdivide	Поэлементное деление массивов слева направо	./

Константа – это предварительно определенное числовое или символьное значение, представленное уникальным именем. Существует четыре типа констант:

1) символьные константы – последовательность символов, заключенных в апострофы;

Таблица 1.3

Операторы отношения

Функция	Название	Оператор
eq	Плюс	==
ne	Унарный плюс	~=
lt	Минус	<
gt	Унарный минус	>
le	Матричное умножение	<=
ge	Поэлементное умножение массивов	>=

Таблица 1.4

Логические операторы

Функция	Название	Оператор
and	Операция «И», логическое умножение	&
or	Операция «ИЛИ», логическое сложение	|
not	Операция «НЕ», инверсия	~
xor	Операция «Исключающее ИЛИ», или сложение по модулю два	**xor**

2) числовые константы – константы, принимающие численные значения. Числа (например, 1, -2 и 1.23) являются безымянными числовыми константами;

3) логические константы – константы, принимающие значения 1 (true) и 0 (false).

4) системные константы. Эти константы в MATLAB принято назвать системными переменными, поскольку, с одной стороны, они задаются системой при ее загрузке, а с другой – могут переопределяться. Основные системные переменные, применяемые в системе MATLAB, указаны ниже в табл. 1.5. Полный список системных констант может быть выведен в результате выполнения команды ***help elmat.***

Переменная – объект языка MATLAB, который может изменять свое значение в процессе вычислений. Существует два типа переменных:

1) простые переменные;

2) массивы.

Существует несколько правил для составления имен переменных:

1) длина произвольная, но значимы только N символов без пробелов (31 по умолчанию);

2) символы имени – латинские буквы, арабские цифры, символ « _ »;

3) начало имени – буква;

4) имя переменной должно быть уникальным;

5) строчные и заглавные буквы различаются.

Таблица 1.5

Основные системные константы

Имя константы	Назначение	Пример
i или ***j***	Мнимая единица	>> i ans = 0 + 1.0000i
pi	Число π	>> pi ans = 3.1416
eps	Погрешность для операций над числами с плавающей точкой	>> eps ans = 2.2204e-016
realmin	Наименьшее число с плавающей точкой	>> realmin ans = 2.2251e-308

realmax	Наибольшее число с плавающей точкой	>> realmax ans = 1.7977e+308
inf	Значение машинной бесконечности	>> 5/0 ans = Inf
nan	Указание на нечисловой характер данных (Not-a-Number)	>> 0/0 ans = NaN

Для задания переменным значений используется операция присваивания, вводимая знаком равенства: ***имя переменной = выражение.***

Функция – объект языка MATLAB со стандартным именем, который предназначен для выполнения действий с несколькими параметрами, которые заключены в круглые скобки и имеет формат:

[Y1, Y2, ...] = <имя функции> (X1, X2, ...),

где *<имя функции>* – стандартное имя функции, *(X1, X2, ...)* – входные параметры, *[Y1, Y2, ...]* – выходные (возвращаемые) параметры, которые часто представляются в виде вектора. Если результат вычислений не присваивается никакой переменной, то он автоматически помещаются в переменную с именем ***ans***. Несколько стандартных математических функций системы MATLAB представлены в табл. 1.6. 12

Выражение – объект языка MATLAB, представляющий собой совокупность констант, функций и переменных, которые объединены символами операций. В системе MATLAB существует три типа выражений: арифметические, логические и символьные.

Таблица 1.6

Математические функции

Функция	Назначение	Пример
abs(a)	Возвращает абсолютную величину для каждого числового элемента вектора a. Если a содержит комплексные числа, *abs(a)* вычисляет модуль каждого	>> a =-5; abs(a) ans = 5 >> a = [-5 7 -6]; abs(a) ans = 5 7 6

	числа	
exp(a)	Возвращает экспоненту для каждого элемента a	>> a = -5; exp(a) ans = 0.0067 >> a = [5 7 6]; exp(a) ans = 1.0e+03 * 0.1484 1.0966 0.4034
factorial(a)	Возвращает произведение всех натуральных чисел от 1 до a включительно	>> a= 5; factorial(a) ans = 120 >> a=[5 6]; factorial(a) ans = 120 720
sqrt(a)	Вычисление квадратного корня из элементов массива a	>> a= 49; sqrt(a) ans = 7 >> a= [49 -49]; sqrt(a) ans = 7.0000 0 + 7.0000i
log(a)	Возвращает значение натурального логарифма элементов массива a	>> a = 2; log(a) ans = 0.6931 >> a= [2 3]; log(a) ans = 0.6931 1.0986
log2(a)	Возвращает значение логарифма по основанию 2 элементов массива a	>> a = 2; log2(a) ans = 1 >> a = [2 3]; log2(a) ans = 1.0000 1.5850
log10(a)	Возвращает значение логарифма по основанию 10 элементов массива a	>> a = 2; log10(a) ans = 0.3010 >> a = [2 3]; log10(a) ans = 0.3010 0.4771
sin(a), asin(a), cos(a), acos(a), tan(a), atan(a), cot(a)	Возвращает значение тригонометрической функции для каждого элемента a (если задан вектор). Аргумент тригонометрической функции задаётся в радианах. Для пересчёта использовать выражение a[рад] = (pi*angle[град]/180)	>> a = 23; sin(a) ans = -0.8462 >> a = [2 3]; sin(a) ans = 0.9093 0.1411

% Примеры выражений:

>> x + sin (a + b – c / d) % Арифметическое выражение

>> a – sqrt (b + c) & (i = = j) % Логическое выражение

>> syms a b c; % Символьное выражение

f = a + b + c

В системе MATLAB приоритет операций регулируют скобки (табл. 1.7). Математическое выражение должно быть записано в одну строку или с использованием многоточия в качестве указателя переноса строки:

y = (log10(8) + log10(3))...*

1/3(log10(16))

Таблица 1.7

Примеры различия записи выражений системы MATLAB от общематематической

Математическая запись	Запись в MATLAB
$-0{,}123 \cdot 10^{-7}$	-0,123e-7
$123 \cdot 4567$	123*4567
$8\pi + e^{31}$	8*pi+exp(31)
$\|x\| + \sqrt{39} + y^7$	abs(x)+sqrt(39)+y^7
$\sin 19 + \sin 19^{\pi}/180$	sin(19)+sin(19*pi/180)
$\log 7 + \log 3$	log(7)+log(3)

Основным типом данных, с которым производятся вычисления в MATLAB является **double**. При этом все числа представляются в виде вещественных чисел с плавающей точкой двойной точности (8 байт). При этом достигается точность порядка *15 значащих цифр*. Такой формат называется **long**. Чтобы не перегружать командное окно для вывода обычно используется формат **short**, при котором на экран выводятся только *4 знака* после запятой. Для того, чтобы вывести вещественные числа в командное окно в формате двойной точности следует набрать команду

format long.

Вернуться в стандартный режим вывода можно командой

format short.

Для работы с целыми беззнаковыми числами можно использовать форматы: **uint8, uint16, uint32,** а для работы с целыми числами со знаком: **int8, int16, int32.** Цифра, которая входит в обозначение формата, показывает число двоичных разрядов, которые используются для представления чисел. Диапазоны используемых целых чисел при ведены в табл. 1.8.

Таблица 1.8

Диапазоны целых чисел

Название	Число бит в представлении числа	Диапазон чисел
uint8	8	0...255
int8	8	-128...+127
uint16	16	0...65 535
int16	16	-32 768...+32 767
uint32	32	0...4 294 967 295
int32	32	-2 147 483 648...+2 147 483 647

Структура файла-сценария обычно имеет следующий вид:

% Основной комментарий (имя программы)

% Дополнительный комментарий (описание программы)

Тело_файла (Содержит любые допустимые выражения MATLAB)

При написании программ следует помнить:

– в языке MATLAB переменные не описываются. Любое новое имя, появляющееся в тексте программы при ее выполнении, воспринимается системой как имя матрицы;

– целесообразно каждый оператор записывать в отдельной строке текста программы. Признаком конца оператора является конец строки. Допускается размещать несколько операторов в одной строке, тогда предыдущий оператор этой строки должен заканчиваться символом « ; » или « , »;

– длинные операторы записываются в несколько строк. При этом предыдущая строка оператора должна заканчиваться тремя точками « ... »;

– если очередной оператор не заканчивается символом « ; », результат его действия при выполнении программы будет выведен в командное окно. Если это не требуется, то вывод на экран результатов можно подавить, завершив оператор символом « ; »;

– в программах MATLAB символ окончания текста программы отсутствует.

Стандартные файлы MATLAB

В MATLAB используются несколько стандартных расширений файлов:

– *m-файлы* (с расширением **.m**) – файлы, содержащие тексты программ на языке MATLAB (файлы-сценарии и файлы-функции);

– *fig-файлы* – файлы с описанием оконных форм (Figure) пользовательского интерфейса, обычно используются в программах на MATLAB;

– *mdl-файлы* – файлы моделей Simulink;

– *p-файлы* – промежуточные файлы, содержащие скомпилированный код процедур MATLAB. Использование p-файлов обычно существенно повышает быстродействие по сравнению с выполнением (интерпретированием) m-файлов и позволяет защитить от несанкционированного доступа программный код;

– *mat-файлы* – файлы, содержащие данные в двоичном коде (обычно значения каких-либо переменных), доступ к которым возможен из командного окна MATLAB либо с помощью специальных средств Simulink.

3. Подготовка к работе

3.1. Ознакомиться с теоретическим материалом и рекомендованной литературой.

3.2. Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

Организовать ввод данных и вычисления согласно заданиям ниже.

Задание 1

Изучить структуру и меню основного окна MATLAB. Определить и записать название и версию используемого программного продукта.

Ввести в командной строке, согласно варианту из табл. 1.10, системные переменные (системные константы) и получить их значения в коротком и длинном форматах представления чисел.

Таблица 1.10

Варианты для выполнения задания 1

№ варианта	Системные переменные
1	2π, eps, realmin, realmax, 1:0, 0:0
2	8π, eps4, realmin, realmax, 2:0, 0:0
3	25π, eps, realmin, realmax, 3:0, 0:0
4	5π, eps2, realmin, realmax, 4:0, 0:0
5	7π, eps7, realmin, realmax, 5:0, 0:0
6	24π, eps, realmin, realmax, 6:0, 0:0
7	π, eps2, realmin, realmax, 7:0, 0:0
8	3π, eps, realmin, realmax, 8:0, 0:0
9	8π, eps4, realmin, realmax, 9:0, 0:0
10	12π, eps, realmin, realmax, 10:0, 0:0

Задание 2

В командном режиме создать 4 простые переменные и присвоить им числовые значения, полученные в результате вычисления арифметических выражений из табл. 1.11. Убедиться в правильности вычислений. Результаты внести в отчет.

Используя встроенный редактор MATLAB, оформить решение указанных выше задач в виде программы и сохранить его в виде m-файла.

№			
1	2 /5+ 7/ 16	sin13 sin13^0	log7+ log3 1 /5
2	0,16 -8/ 15	ctg16-tg45^0	Log_2 7+ lg3 2/ 7

3	8 /13 +0,58	sin18+ $tg16^0$	lg30- $\log_2$ 8 3/7
4	0,5- 3 /8	ctg18+ $\cos16^0$	lg81+ $\log_2$ 0,9 13 /5
5	5 /24 +7,13	ctg18+ $\sin18^0$	$\log_2$2+ lg10 13/ 7
6	3/ 11 -3,19	tg30+$tg30^0$	$\log_2$ 19+ lg19 8 /3
7	6/15- 0,15	ctg19-$tg16^0$	lg7 +lg0,8 12 /7
8	5 /12+ 5 /7	ctg18+$tg18^0$	lg49 $\log_2$ 8 1 /8
9	0,16- 6,15	cos15 -$\cos15^0$	$\log_2$2+ $\log_2$ 8 3/ 5
10	5,1 +8/ 13	sin10- $\sin10^0$	$Log_2$10 +$\log_2$ 2 3/ 5

5. Требования к отчёту

Отчёт должен содержать:

– результаты выполнения задания, сохраненное в файле, имеющем имя в виде фамилии студента.

6. Контрольные вопросы

6.1. Из каких частей состоит основное окно системы MATLAB? Дайте краткое описание каждой из этих частей.

6.2. Что представляет собой режим прямых вычислений? Режим команд?

6.3. Что такое текущая сессия? Как просмотреть список переменных, созданных в текущей сессии?

6.4. Какие основные правила должны соблюдаться при вводе команд в командную строку?

6.5. Назовите основные объекты языка MATLAB.

6.6. Что такое команда языка MATLAB? Опишите синтаксис команды HELP.

6.7. Что такое оператор? Какие виды операторов существуют?

6.8. Что такое константа? Назовите основные системные константы MATLAB. Какие типы констант существуют?

6.9. Что такое переменная? Назовите основные правила для составления имен переменных.

MUHAMMAD AL... NOMIDAGI
TOSHKE... AXBOROT
TEXNOLOGIYA... UNIVERSITETI
AXBOROT-RESURS MARKAZI

6.10. Какая операция используется для задания переменным значений? Опишите синтаксис этой операции.

6.11. В какой переменной хранится возвращаемое функцией значение?

6.12. Что такое функция и каков ее формат?

6.13. Назовите основные математические функции, используемые в системе MATLAB.

6.14. Что такое выражение в языке MATLAB? Назовите основные типы выражений.

6.15. Какая команда позволяет получить сведения о переменных в рабочем пространстве?

6.16. Какая команда позволяет удалить переменные из памяти?

6.17. Что представляет собой комплексное число?

6.18. Какие основные функции используются для работы с комплексными числами?

6.19. Как выполнить очистку окна Command Windows?

6.20. Какую функцию выполняет точка с запятой в конце команды?

6.21. Как можно использовать встроенный редактор MATLAB для создания и отладки программ?

6.22. Как организовать программу на языке MATLAB в виде m-файла?

6.23. Какие типы файлов используются в MATLAB?

ЛАБОРАТОРНАЯ РАБОТА № 2
РАБОТА С ГРАФИКАМИ В СИСТЕМЕ MATLAB

1. Цель работы

Изучение команд построения графиков в различных системах координат с целью визуализации результатов вычислений. Получение практических навыков работы в командном и диалоговом режиме при форматировании внешнего вида графиков.

2. Теоретический материал

В результате вычислений в системе MATLAB зачастую получается большой, трудно анализируемый массив данных. Для наглядной визуализации и анализа этих данных в системе MATLAB предусмотрен специальный встроенный пакет. Рассмотрим наиболее простые и часто используемые графические возможности.

Графики в декартовой системе координат Графические объекты, в том числе и графики функций, MATLAB выводит в специальные графические окна, имена которых обозначаются словом **Figure**. Для построения графиков функций в декартовой системе координат служит команда *plot(x, y)*, где координаты точек (x, y) берутся из векторов X и Y соответственно. Отсюда следует, что перед использованием функции построения графиков *plot(x, y)* эти вектора должны быть сформированы. Например: *% Построить график функции y=sin(x)-cos(x).*

>> x=0: 0.1:10; % задаем диапазон и шаг изменения % аргумента x от 0 до 10 с шагом 0,1

>> y=sin(x)-cos(x); % задаем функцию, которую нам % необходимо построить

>> plot(x, y) % строим график функции в графическом окне При этом в графическом окне автоматически строятся две оси с нанесенными значе-ниями функции и аргумента: *абсцисс* (ось x) и *ординат* (ось y); задаются координаты (x, y), которые определяют узловые точки функции *y(x)*. Результат выполнения команд изображен на рис. 2.1.

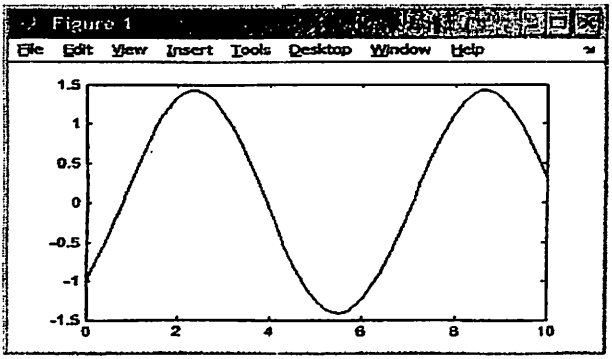

Рис. 2.1 – График функции *y=sin(x)-cos(x)*

Аналогичная команда ***plot(x, y, s)*** позволяет построить график функции, указав способ (стиль) отображения линии и узловых точек при помощи строковой переменной ***s***, которая может включать до трех символов маркеров из табл. 3.1. Строковую переменную *s* следует задавать в апострофах, как показано в примере ниже, причем порядок указания маркеров не имеет значения. Отсутствующие маркеры используют установки по умолчанию. При отсутствии указания на цвет линий и точек он выбирается автоматически (белый цвет исключается). Если линий графиков больше шести, то выбор цвета повторяется.

% Построить график функции заданным стилем

>> x=0:0.25:10; % задаем распределение значений x

>> y=sin(x)-cos(x);

% задаем функцию, которую необходимо построить

>> plot (x,y,'-hk') % строим график функции в виде непрерывной линии

% чёрного цвета с шестиугольными фигурами

% в контрольных точках

Результат выполнения команд изображен на рис. 2.2.

Таблица 2.1

Типы и цвета линий и узловых точек

Тип линии		Тип точки		Цвет	
Вид	Маркер	Вид	Маркер	Вид	Маркер
Непрерывная	-	Точка	.	Жёлтый	y
Штриховая	--	Плюс	+	Фиолетовый	m
Двойная пунктирная	:	Звёздочка	*	Голубой	c
Штрих-пунктирная	-.	Круг	o	Красный	r
		Крест	x	Зелёный	g
		Квадрат	s	Синий	b
		Ромб	d	Белый	w
		Треугольник (вниз)	v	Чёрный	k
		Треугольник (вверх)	a		
		Треугольник (влево)	<		
		Треугольник (вправо)	>		
		Пятиугольник	p		
		Шестиугольник	h		

Команда *plot(x1, y1, s1, x2, y2, s2, ...)* служит для объединения на одном графике нескольких графиков функций, определив для каждой свой стиль отображения.

% Построить в одном окне графики функций y1=sin(x1)-cos(x1)

% и y2=tg(x2). >> x1=0:0.1:2;

>> x2=0:0.2:2;

>> y1=sin(x1)-cos(x1);

>> y2=tan(x2);

>> plot(x1,y1,'-hm',x2,y2,'--sb')

Результат выполнения команд изображен на рис. 2.3.

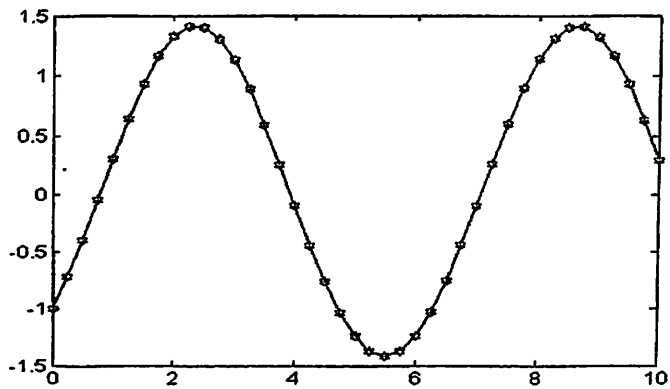

Рис. 2.2 – График функции *y=sin(x)-cos(x)* со спецификацией точек и линии

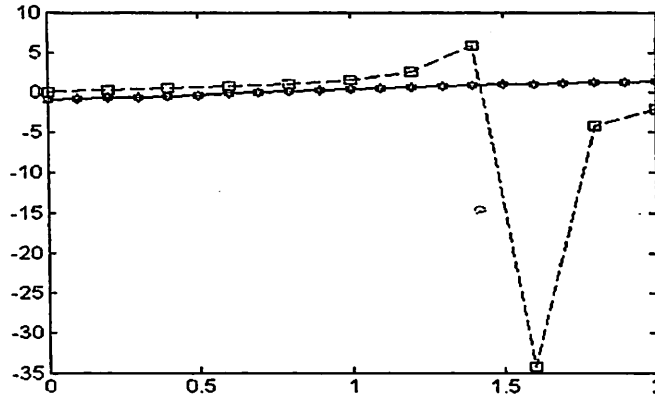

Рис. 2.3 – Графики функций *y1=sin(x1)-cos(x1)* и *y2=tg(x2)*

Графики в полярной системе координат

В полярной системе координат любая точка представляется как конец радиус-вектора, исходящего из начала системы координат, который имеет длину **rho** и угол **phi**. Для построения таких графиков используются команды:

polar(phi, rho),

polar(phi, rho, s),

где *phi* – угол, *rho* – длина радиус-вектора, *s* – строковая константа, позволяющая задавать стиль построения графиков, значения которой берутся из табл. 2.1.

% Построить в полярной системе координат график функции r=cos(6φ).

*>>phi=0: pi/60: 2*pi;*

*>> r = cos(6*phi); polar(phi, r)*

Результат выполнения команд изображен на рис. 2.4.

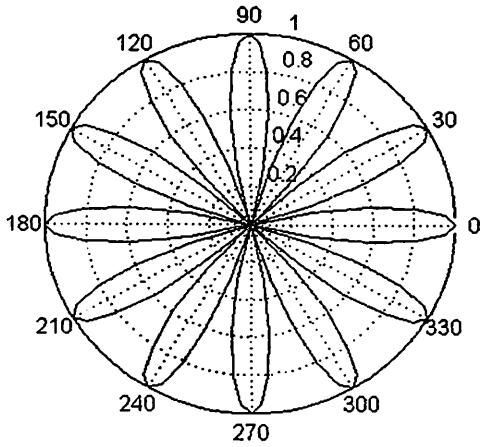

Рис. 2.4 – График функции *r=cos(6φ)* в полярных координатах

Графики векторов

Для представления *комплексных элементов одномерного массива Z* в виде векторов-стрелок, исходящих из начала координат и имеющих угол и длину, которые определяются действительной и мнимой частью комплексных чисел, служит команда

compass (Z).

Команда ***compass (real(Z), imag(Z))*** эквивалентна команде ***compass (real(Z) + imag(Z))***. Здесь *real(Z)* – действительная часть комплексного числа, *imag(Z)* – мнимая часть комплекс-ного числа;

При необходимости задать тип линии для отображения векторов используется дополнительный аргумент *s*, значения которого берутся из табл. 2.1. ***compass (Z, s).***

% Построить радиус-векторы трех комплексных чисел 4i-6, 6i+8 и 7i-13.

*>> x=[4*i-6;6*i+8;7*i-13]; % задаем массив комплексных чисел*

>>compass(x) % выводим радиус векторы в графическое окно

Результат выполнения команд изображен на рис. 2.5.

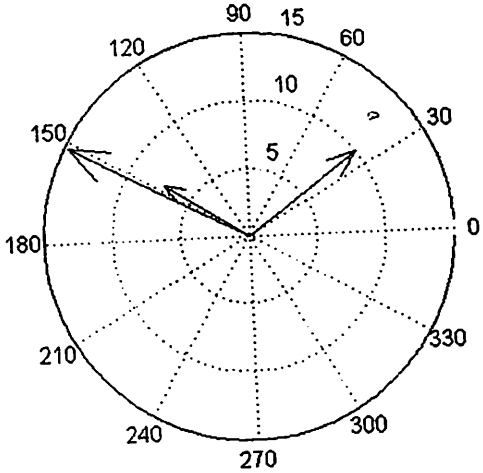

Рис. 2.5 – Радиус-векторы комплексных чисел *4i-6*, *6i+8* и *7i-13*

Трехмерные графики

Простейшей функцией построения графика функции двух переменных *z=f(x,y)*, который представляет собой поверхность, является

plot3(X, Y, Z),

где *X* – матрица первых координат сетки точек, *Y* – матрица вторых координат, *Z* – матрица значений функции двух переменных. Эта функция строит в трехмерных координатах точки с координатами *(x, y, z)* и соединяет их отрезками прямых. По аналогии с функцией ***plot*** можно использовать строковую переменную *s* с указанием способа отображения линии и узловых точек. Перед использованием функции ***plot3*** необходимо сформировать матрицы значений координатной сетки *X* и *Y*. Для их получения используется функция

[X, Y] = meshgrid(x, y).

Эта функция получает два одномерных массива (вектора) точек на осях *x* и *y*, а возвращает два двумерных массива *X* и *Y*. Функция *[X,Y]=meshgrid(x)* представляет собой упрощенную запись для *[X,Y]=meshgrid(x, x)*. Нанести на систему координат шкалы в виде сетки можно командой ***grid on***.

% Построить график 3D-поверхности, описываемой функцией $z(x,y)=x^2+y^3$.

>>[X,Y]=meshgrid([-3:0.5:5]); % формируем прямоугольную сетку координат

>> Z=X.^2+Y.^3; % задаем функцию

>>plot3(X,Y, Z); % построить поверхность

>> grid on % включить отображение шкал в виде сетки

Результат выполнения команд изображен на рис. 2.6. Чем меньше будет шаг координатной сетки, тем более сглаженной будет казаться поверхность.

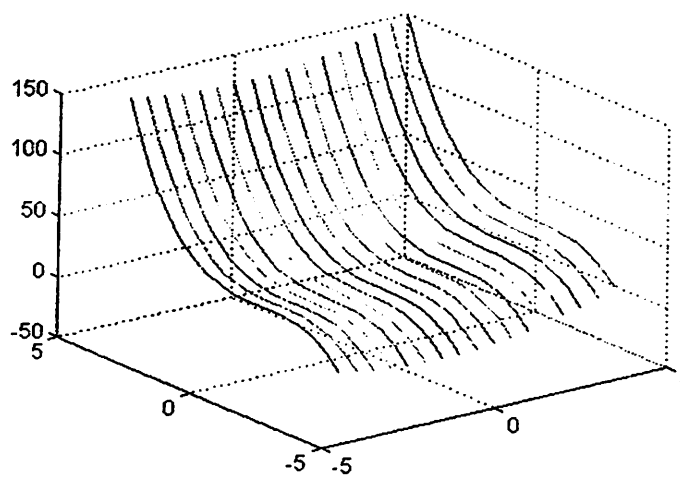

Рис 2.6 – График 3D-поверхности, описываемой функцией $z(x, y)=x^2+y^3$

Более наглядными являются сетчатые графики поверхностей с заданной или функциональной окраской. Для построения сетчатой поверхности, цвет

которой меняется в зависимости от значения Z, используется функция ***mesh(z)***. При этом координатная число узлов сетчатой поверхности определяется размерностью массива Z.

% Построить график функци z(x, y)=x2+y3.

>>[X,Y]=meshgrid([-3:0.5:5]); % формируем прямоугольную сетку координат

```
>> Z=X.^2+Y.^3;                % задаем функцию
>>mesh(Z)                     % построить сетчатую поверхность
>>colormap copper             % задаём палитру графика
```

Результат показан на рис. 2.7. Особенно наглядное представление о поверхностях дают сетчатые графики, использующие функциональную закраску ячеек. Такие графики можно построить используя команду ***surf(X,Y,Z)***.

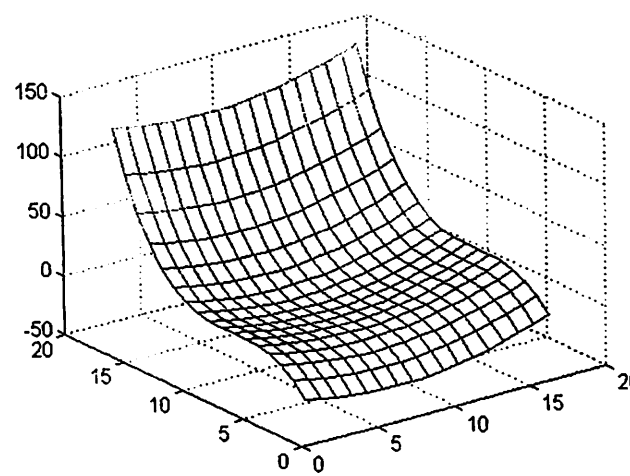

Рис. 2.7 – Сетчатый график 3D-поверхности, описываемый функцией $z(x,y)=x^2+y^3$

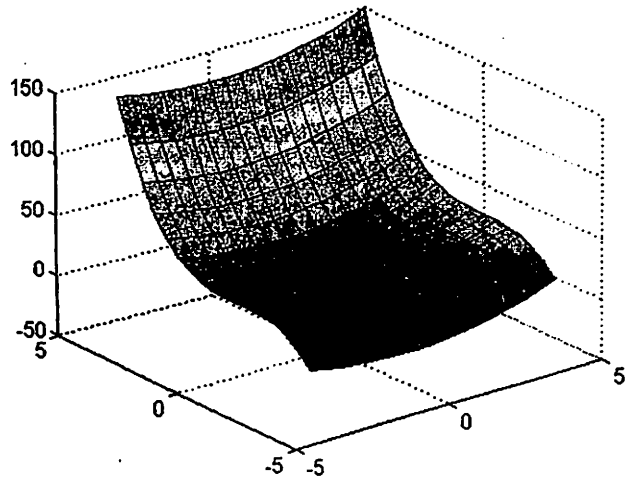

Рис. 2.8 – Сетчатый график 3D-поверхности с функциональной окраской ячеек

Оформление графиков

Для наглядной визуализации данных и возможности использования их в различных отчётах, в большинстве случаев, не достаточно только вывести данные в виде графиков на экран, а требуется придать им определённый вид. Настроить внешний вида графиков в среде MATLAB можно используя команды графики или инструменты меню графического окна.

Для масштабирования графиков функций используется команда

axis([Xmin, Xmax, Ymin, Ymax, Zmin, Zmax]).

В данной команде также можно использовать ключевые слова для задания внешнего вида осей (см. табл. 2.2).

Команды для отображения сетки на графике, создания подписей и добавления легенды представлены в табл. 2.3.

Для оформления трёхмерных графиков, кроме команд указанных в табл. 2.3, можно использовать команды изменяющие окраску и вид поверхности (табл. 2.4).

Таблица 2.2

Команды управления осями

Название команды	Действие
axis square	Делает оси *x* и *y* равной длины
axis equal	Создаёт отдельные отметки приращений для осей *x* и *y* одинаковой длины
axis auto	Возвращает значения осей по умолчанию и переходит в автоматический режим
axis [on, off]	Включает/выключает обозначения осей и меток промежуточных делений

Таблица 2.3

Команды управления осями

Название команды	Действие
grid [on, off]	Включает / выключает сетку на графике
title ('заголовок графика')	Создаёт надпись заголовка графика
xlabel ('подпись оси Ox')	Создаёт подпись оси *Ox*
ylabel ('подпись оси Oy')	Создаёт подпись оси *Oy*
text (x, y, 'текст')	Создаёт текстовую надпись в

	координатах *(x, y)* графического окна
legend('подпись выводимой функции №1', 'подпись выводимой функции №2', ...)	Добавляет легенду к текущему графику

Таблица 2.4

Команды управления осями

Название команды	Действие
shading flat	Удаляет сетку на поверхности
colormap([палитра])	Задаёт окраску поверхности
colorbar	Выводит вертикальную шкалу цветов текущего графика
view (az, el)	Изменяет угол наблюдения графика (az – азимут, el – угол возвышения)

% Оформить график функции z(x, y)=x2+y2 . используя доп. команды

>>[X,Y]=meshgrid([-3:0.5:5]); % формируем сетку координат

>>Z=X.^2+Y.^3; % задаем функцию

>>surf(X,Y, Z); % строим 3D-график функции

>>axis([-3, 5, -3, 5, -30, 150]); % изменяем масштаб осей

>>title('График функции z=x^2+y^3'); % добавляем заголовок графика

>>xlabel('X'); ylabel('Y'): zlabel('Z', 'Rotation', 0);

% подписываем оси координат

>>colorbar; % выводим шкалу цветов

Результат работы программы показан на рис.2.9.

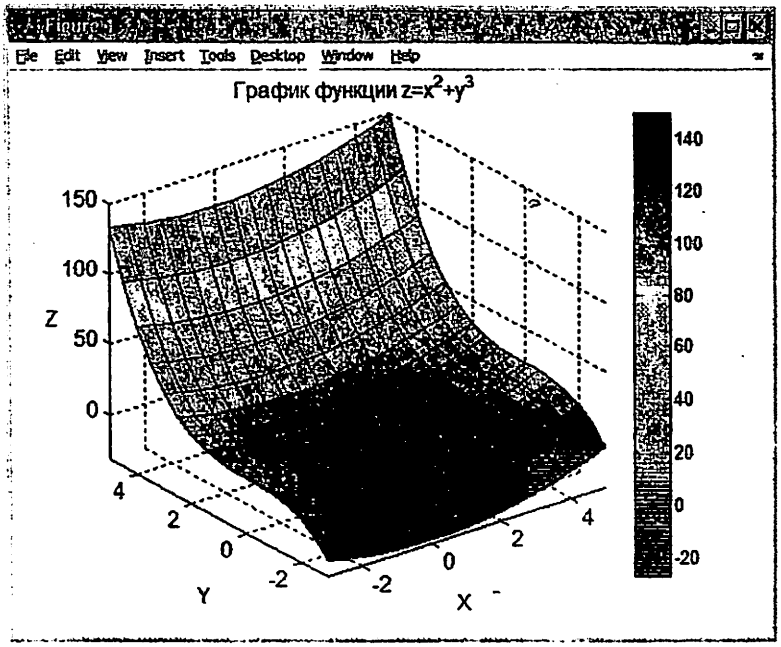

Рис. 2.9 – Оформленный график 3D-поверхности

Второй способ форматирования вида графика заключается в использовании инструментов графического окна. Так из меню *Insert* графического окна доступно добавление легенды, подписей графика и осей, текстовых блоков, сетки графика, шкалы цветов, а в случае, если необходимо отредактировать вид отдельных элементов графика выбрать в меню *View* пункт *Property Editor* и выбрать мышкой интересуюший элемент графика.

Построение графиков разного типа в одном окне

Если необходимо построить в одном графическом окне *figure* несколько координатных осей с различными графиками без наложения их друг на друга, то можно воспользоваться функцией ***subplot(m, n, p)***, где m – число подокон по горизонтали, n – число подокон по вертикали, а p – номер подокна, в которое

будет выводиться текущий график функции (подокна отсчитываются последовательно по строкам).

```
% Построить график функции z(x, y)=x^2+y^2 разными
% командами в одном окне
>>[X,Y]=meshgrid([-3:0.5:5]);          % формируем сетку координат
>>Z=X.^2+Y.^3;                         % задаем функцию
% строим 3D-графики функции в одном масштабе осей
>>subplot(2, 2, 1), plot3(X, Y, Z), axis([-5, 5, -5, 5, -50,150]);
>>subplot(2, 2, 2), mesh(X, Y, Z), axis([-5, 5, -5, 5, -50,150]);
>>subplot(2, 2, 3), surf(X, Y, Z), axis([-5, 5, -5, 5, -50,150]);
```

Результат работы программы показан на рис. 2.10.

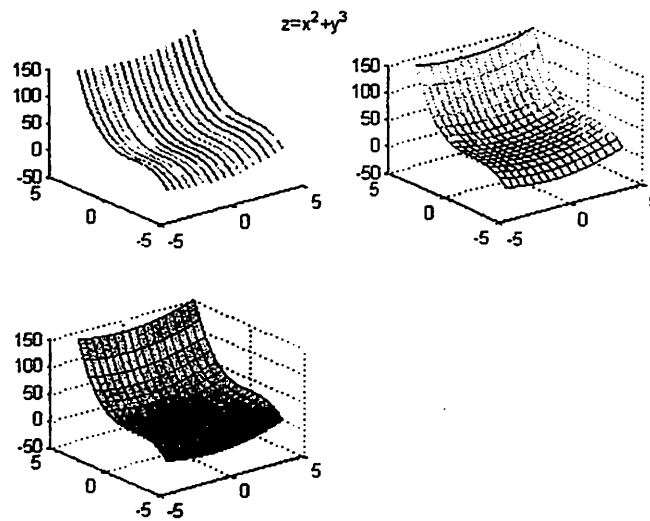

Рис. 2.10 – Вывод несколько графиков в одном окне

3. Подготовка к работе

3.1. Ознакомиться с теоретическим материалом и рекомендованной литературой.

3.2. Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

Организовать ввод данных и вычисления согласно заданиям ниже.

Задание 1

Создать и отладить программу (в виде m-файла) для построения трех графиков функций в одной системе координат, самостоятельно задав шаг и диапазон изменения аргумента. Функцию выбрать из таблицы 2.5 согласно варианту.

Таблица 2.5

Варианты для выполнения задания 1

№ варианта	Функции	№ варианта	Функции
1	$y= x^2-4$ $y= x^4-4\ x^2$ $y= \sin(x)+ \log(x)$	6	$y= 3x^4+4x^3$ $y= 2x^4- x^2$ $y= \sin(x)- \sin(4x)$
2	$y= x^3-3$ $y= x^4-14x$ $y= tg(x)+ lg(x)$	7	$y= 8-4x+x^3$ $y= x^2-3x^4$ $y= \sin(x)+ \log(5x)$
3	$y= x^2-16$ $y= 5x^3-8x$ $y= \cos(x)+ \sin(x)$	8	$y= x^2-5$ $y= x^4-8x^2$ $y= \sin(4x)+ \log(2x)$
4	$y= x^2-5x-4$ $y= 2x^4-4\ x^3$ $y= \sin(x)+4\sin(2x)$	9	$y= 3x^2-3$ $y= 4x^4-4\ x^2$ $y= \cos(x)+ tg(x)$
5	$y= x^3+4$ $y= x^4+7x$ $y= \arcsin(x)+ ctg(x)$	10	$y= 4x^2-4$ $y= x^4- x^3$ $y= tg(x)+ ctg(3x)$

5. Требования к отчёту

Отчёт должен содержать:

– титульный лист с указанием названия ВУЗа, кафедры, номера и темы лабораторной работы, а также фамилии И.О. студента, подготовившего отчёт;

– цель выполняемой работы;

– задания;

– листинги всех программ с обязательными комментариями;

– полученные на каждом этапе работы результаты (графики);

– выводы по каждому выполненному заданию.

6. Контрольные вопросы

6.1 В окно с каким именем выводятся построенные графики функций?

6.2 Назовите команды для построения графиков в декартовой системе координат. Что является аргументами этих команд?

6.3 Каким образом можно изменить стиль графика?

6.4 Как объединить в одной системе координат несколько графиков функций?

6.5 Для чего используется полярная система координат?

6.6 Какие команды используются для построения графиков в полярной системе координат?

6.7 Для чего служит и как применяется функция *compass*?

6.8 Какие функции MATLAB используются для построения трехмерных графиков функций?

6.9 В чем заключается особенность применения функции *plot3*?

6.10 Как можно построить 3D график в виде сетчатой поверхности?

6.11 Как можно построить 3D график в виде сетчатой поверхности с функциональной окраской?

6.12 Каким способом можно вывести в одно графическое окно несколько разных графиков?

6.13 Какие команды MATLAB используются для изменения вида и стиля графиков?

6.14 Какие интерактивные инструменты используются для изменения вида и стиля графиков?

6.15 Какие правила следует выполнять при подготовке программ в виде m-файлов?

6.16 В чём отличия графиков построенных командами mesh(Z) и mesh(X,Y,Z)?

ЛАБОРАТОРНАЯ РАБОТА №3
ПРОСТЕЙШИЕ ОПЕРАЦИИ С ИЗОБРАЖЕНИЯМИ

1. Цель работы

Изучение возможностей пакета Image Processing Toolbox. Изучение основных приёмов работы с изображениями. Получение практических навыков в составлении программ.

2. Теоретический материал

Пакет **Image Processing Toolbox (IPT)** представляет собой набор функций для расширения возможностей системы MATLAB при работе с цифровыми изображениями. Данный пакет поддерживает различные операции для обработки изображений, такие как:

1) пространственное преобразование;

2) нелинейная и линейная фильтрация;

3) анализ и улучшение изображений;

4) восстановление изображений;

5) сжатие изображений.

Представление цифровых изображений ***Монохромное*** (черно-белое) изображение (Grayscale) можно определить как двумерную функцию $f(x, y)$, где

x и *y* представляют пространственные координаты, а амплитуда *f* для каждой пары координат (x, y) является интенсивностью или яркостью изображения в точке с заданными координатами. ***Цветные*** изображения формируются комбинацией нескольких монохромных. Например в системе **RGB** цветное изображение состоит из комбинаций трех монохромных компонент **R** - красной, **G** - зеленой и **B** - синей. В ***аналоговых изображениях***, например, изображениях на фотографиях, или на картинах, нарисованных художником, каждую точку можно характеризовать непрерывными пространственными координатами (x, y) и соответствующей интенсивностью или цветом. Преобразование аналогового изображения в цифровое требует представления координат и интенсивностей некоторыми дискретными значениями. Для ***оцифровки*** выполняются операции:

– ***пространственная дискретизация координат***, при которой непрерывные координаты заменяются конечным множеством отсчетов;

– ***квантование по амплитуде***, при котором интенсивности в каждой полученной точке заменяются квантованными значениями (округленными до некоторых эталонных).

Результатом дискретизации и квантования является *матрица чисел*, состоящая из *элементов изображения* (*пикселов*) и имеющая **M** строк и **N** столбцов. Это значит, что изображение имеет размер **MxN**. ***Цифровые изображения***, загруженные в среду MATLAB, помещаются в прямоугольную систему координат, где дискретные координаты (x, y) заменяются на (r, c) для обозначения строк (*row*) и столбцов (*column*). Начало этой системы соответствует верхнему левому углу изображения, координаты которой $(r, c) = (1, 1)$. При этом значения координат могут меняться следующим образом $r = 1, 2, \ldots, M$; $c = 1, 2, \ldots, N$. Математически такое цифровое изображение описывается матрицей (3.1), вид которой приведен ниже.

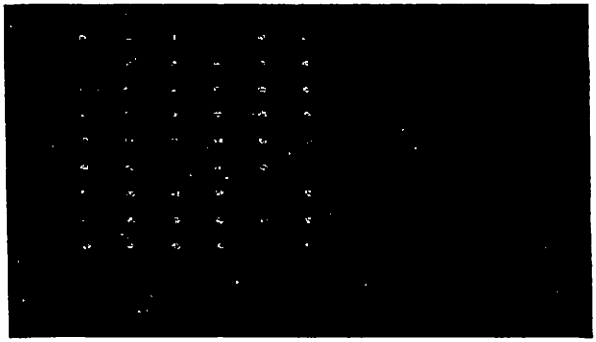

Рис. 3.1 – Координатное соглашение в системе MATLAB .

$$f(r,c)= \begin{bmatrix} f(1,1) & f(1,2) & \dots & f(1,N) \\ f(2,1) & f(2,2) & \dots & f(2,N) \\ \dots & & & \\ f(M,1) & f(M,2) & \dots & f(M,N) \end{bmatrix}$$

Классы данных

Координаты пикселов изображений являются целыми числами, однако *значения пикселов* (интенсивность, яркость), несмотря на то, что они проквантованы, могут принимать дискретные значения *в разных диапазонах числовых значений*. Это определяется *классом данных*, который используется для представления изображений. Используемые в MATLAB классы данных приведены в табл. 3.1.

Таблица 3.1

Классы данных

Имя класса	Описание
uint8	Целые числа без знака в интервале [0, 255] (1 байт на число)
uint16	Целые числа без знака в интервале [0, 65 535] (2 байта на число)
uint32	Целые числа без знака в интервале [0, 4 294 967 295] (4 байта на число)
int8	Целые числа со знаком в интервале [-128, 127] (1 байт на число)
int16	Целые числа со знаком в интервале [-32 758, 32 767] (2 байта на число)
int32	Целые числа со знаком в интервале [-2 147 483 648, 2 147 483 647] (4 байта на число)
single	Вещественные числа с плавающей запятой одинарной точности в диапазоне, примерно, от -10^{38} до 10^{38} (4 байта на число)
double	Вещественные числа с плавающей запятой двойной точности в диапазоне, примерно, от -10^{308} до 10^{308} (8 байт на число)
char	Символы (буквы и знаки) (2 байта на символ)
logical	Значения логического 0 или 1 (1 байт на элемент)

Классы данных **double, uint8** и **logical** (см. табл. 4.1) наиболее часто используются для представления изображений. Класс **uint16** используется значительно реже. Символьный класс **char** хранит символы в кодировке Unicode. Символьная строка представляет собой *массив размером 1xn* символов, в котором каждый символ занимает *2 байта*. Класс данных **logical** используется для описания логических массивов, элементы которых содержат значения логических 0 или 1. Такие массивы хранятся в памяти занимая по 1

байту на пиксел. Все численные операции в MATLAB выполняются с двойной точностью в классе **double**. *Типы изображений* Пакет расширений IPT может работать с изображениями следующих типов:

– ***полутоновые*** (черно-белые, монохромные, grayscale) изображения. Это матрица, элементы которой представлены числовыми значениями.

Если элементы этого изображения принадлежат классу **uint8** или **uint16**, то они представляются целыми числами в интервалах [0, 255] и [0, 65 535] соответственно. Если изображение использует класс **double**, то интенсивность пикселов представляется вещественными числами с плавающей точкой двойной точности (см. табл. 3.1). При этом условлено, что интенсивность пикселов лежит в диапазоне [0, 1].

– ***двоичные*** (бинарные, bitmap) изображения являются *логическими массивами*, со-стоящими из **1** и **0**. Массив элементов из 1 и 0 других классов, например, **uint8**, не является двоичным изображением. Для преобразования числовых массивов в логические используется функция *logical*:

G = logical(A),

где *A* – числовой массив, состоящий из 1 и 0, *G* – логический массив с теми же элементами. Если массив *A* имеет элементы отличные от 0 и 1, то эта функция *преобразует все его отличные от 0 элементы в логические 1, а все нулевые в логический 0.*

– ***индексированные цветные изображения***;

– ***полноцветные изображения***.

Таким образом, изображения, обрабатываемые в MATLAB, характеризуются *типом изображения* и *классом данных*. Например: полутоновое изображение класса **double** или бинарное изображение класса **logical**.

Конвертирование классов данных и типов изображений

Преобразование (конвертирование) изображений из одних классов и типов в другие классы и типы является самым распространенным действием в

IPT. Применяя конвертирование типов данных, следует помнить о диапазонах значений величин каждого класса, приведенных в табл. 4.1. Классы конвертируются командой следующего формата: ***B = data_class_name(A)***, где *data_class_name* – это одно из имен классов данных из первого столбца табл. 3.1. Например, пусть *A* — массив класса **uint8**. Массив двойной точности *B* генерируется командой ***B = double (A)***. Такое преобразование используется весьма часто, поскольку MATLAB предполагает, что операнды числовых операций являются вещественными числами с двойной точностью.

Если *C* – это массив класса **double**, элементы которого лежат в интервале [0, 255], но среди них могут встречаться дробные числа, то для их исключения его можно преобразовать в массив **uint8** следующим образом: ***D = uint8(C)***. Если массив класса **double** имел элементы со значениями вне интервала [0, 255] и он был конвертирован в класс **uint8** описанным выше способом, то MATLAB преобразует все отрицательные величины (меньшие 0) в **0**, все величины, большие 255 — в **255**, а у всех остальных элементов *отбрасываются дробные части*. Это означает, что перед преобразованием массивов **double** в **uint8** необходимо совершить подходящую перенормировку (перемасштабирование) его элементов. Преобразование любых числовых данных в логические приводит к логическому массиву, в котором стоят *логические 1* везде на месте ненулевых элементов входного массива и *0* на месте тех элементов, которые равны нулю.

Уточнить информацию о размерах массива и классе изображения *f* можно с помощью команды ***whos f***:

>> whos f Name Size Bytes Class f 3x3 72 double array

В IPT имеются специальные функции, которые реализуют ***перенормировку*** (перемасштабирование) при конвертировании одних классов и типов изображений в другие. Функция ***im2uint8*** сначала распознает класс данных на входе и совершает все необходимые преобразования, чтобы выходное изображение имело правильный тип данных. В качестве примера

рассмотрим изображение *f* размера 2x2 класса **double**, которое может являться результатом некоторых промежуточных вычислений: *f = -0.5 0.5 0.75 1.5* Выполнив преобразование >> *g = im2uint8(f)* получим результат *g = 0 128 191 255* Из примера видно, что функция ***im2uint8*** обнуляет все отрицательные значения входного изображения, ставит число *255* на место величин, больших *1*, и умножает остальные значения на *255*, после чего округляет результат до *ближайшего целого числа*. Преобразование произвольных массивов **double** в пере нормированные массивы (изображения) **double** со значениями в интервале [0, 1] выполняется с помощью функции ***mat2gray***, имеющий следующий синтаксис:

g = mat2gray(A, [Amin, Amax]) ,

где изображение *g* имеет значения пикселов в интервале от *0* (черный) до *1* (белый). Это происходит следующим образом:

– все элементы, меньшие или равные *A*min, обнуляются;

– все элементы, большие или равные *A*max, заменяются на *1,0*.

– остальные элементы отображаются пропорциональными значениями в интервале [0, 1.0].

```
>>f =
1  2
3  4
>> g=mat2gray(f, [1, 4])
g =
0         0.3333
0.6667    1.0000
```

При выполнении команды без параметров *[A*min, *A*max*]*

g = mat2gray(A),

создается новый массив класса **double** в котором значения *A*min и *A*max – это настоящие максимум и минимум массива *A*.

```
>>f1 = 0  3  2
```

```
    6   0   2
    5   0   4
>> g1=mat2gray(f1)
g1 =
0         0.5000   0.3333
1.0000    0        0.3333
0.8333    0        0.6667
```

Функция ***im2double*** преобразует входной массив классов **logical, uint8** или **uint16** в класс **double** с диапазоном **[0, 1]**. Если входной массив был класса **double**, то функция ***im2double*** оставляет его без изменений.

```
>> f2=uint8([15 3 25; 66 0 200; 51 40 49])
f2 =
15   3   25
66   0   200
     51   40   49
>> g2=im2double(f2)
      g2 =
 0.0588    0.0118   0.0980
0.2588    0         0.7843
0.2000 0.1569 0.1922
```

Из примера видно, что входной массив класса **uint8** пере масштабируется делением каждого его элемента на *255*. В случае перемасштабирования массива класса **uint16** каждый элемент будет делиться на *65535*. Для получения двоичных изображений из полутоновых в IPT используется функция ***im2bw***, имеющая синтаксис

```
g = im2bw(f, T)
```

Результатом будет двоичное изображение *g*, полученное из полутонового изображения *f*, преобразованное по порогу *T*. Значения всех элементов *f*, меньших *T*, становятся логическими *0*, а все остальные – логическими *1*. Значение порога *T* должно находиться в интервале [0, 1] независимо от класса входного изображения. Выходной массив автоматически будет логическим. Если порог не указывать

g = im2bw(f),

то по умолчанию он принимается *T = 0.5*.

Если входное изображение было класса **uint8**, то ***im2bw*** сначала делит его элементы на *255*, а потом применяет заданный порог или порог, принятый по умолчанию. Если входной массив был класса **uint16**, то деление производится на *65535*. Если входное изображение принадлежало классу **double**, то ***im2bw*** сразу применяет соответствующий порог.

Если входной массив был логическим, то выходной массив будет ему идентичен. Логический (двоичный) массив можно преобразовать в числовой с помощью любой из четырех функций:

– ***im2uint8***;

– ***im2uint16***;

– ***mat2gray***;

– ***im2double.***

3. Подготовка к работе

3.1. Ознакомиться с теоретическим материалом и рекомендованной литературой.

3.2. Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

Организовать ввод данных и вычисления согласно заданиям ниже. Исходные изображения для выполнения работы хранятся в папке Images_3 методических указаний.

Задание 1

1) Загрузить в рабочее пространство MATLAB **изображение 1** из файла, указанного в таблице с вариантами заданий.

2) Используя функцию ***whos*** определить тип и класс загруженного изображения. Записать результат.

3) Создать и отладить программу, решающую следующие задачи:

а) загрузить изображение с помощью функции ***imread***. Если было выяснено, что изображение цветное (состоит из трех компонент), преобразовать его в полутоновое с помощью функции ***rgb2gray***;

б) с помощью функции ***size*** определить пиксельный размер изображения в виде вектора *[M, N]*;

в) определить максимальное *I*max и минимальное *I*min значения яркостей пикселов изображения;

г) вывести загруженное изображение в графическое окно с помощью функции ***imshow***, при этом сформировать пояснительные надписи: **Original, MxN, Imax=xxx, Imin=xxx**. Для того, чтобы не изменять исходное изображение при выводе на экран следует использовать функцию ***imshow*** без параметров.

д) записать изображение на диск в рабочую папку, используя функцию ***imwrite (f, 'filemame')***, присвоив ему имя **zad3_1** сохранив прежнее расширение.

4) Модернизировать отлаженную программу, оформив решение задач а), б) и в) в виде М-функции, которая должна вызываться из программы.

5) Проанализировать изображение, выведенное в графическое окно, с помощью встроенного инструментария:

а) с помощью инструмента **Data Cursor** нанести на изображение информацию о четырех его угловых точках;

б) с помощью того же инструмента найти и нанести на изображение информацию о *пикселах с максимальной и минимальной яркостью*;

в) сохранить изображение из графического окна, вместе с нанесенными надписями, че-рез меню **File\Save As**, присвоив ему имя **zad3_2** и сохранив прежнее расширение.

6) Загрузить сохраненные изображения из окна Current Directory в просмотрщик графических файлов, используя опцию **Open Outside MATLAB** и проанализировать их. Установить и записать отличия.

Таблица 3.2

Варианты для выполнения заданий

№	Координаты начала фрагмента (r, c)	Размер фрагмента MxN	Коэффициент уменьшения размера изображения		Изображение
			по строкам	по столбцам	
1.	80,80	100x100	2	2	number1
2.	100,150	120x120	2	3	number2
3.	150,150	140x140	3	2	number3
4.	100,100	160x160	3	3	number4
5	135,135	90x90	2	2	number5
6	120,100	150x150	3	3	number6
7	200,160	120x120	3	2	number7
8	145,160	110x110	2	2	number8
9	120,150	120x120	2	3	number9
10	60,100	130x130	3	3	number10
11	110,100	100x100	2	2	number11

12	100,120	120x120	2	3	number12
13	80,80	140x140	3	2	number13
14	100,150	160x160	3	3	number14
15	150,150	90x90	2	2	number15
16	100,100	150x150	3	3	number16
17	135,135	120x120	3	2	number17
18	120,100	110x110	2	2	number18
19	200,160	120x120	2	3	number19
20	145,160	130x130	3	3	number20

Задание2

Создать и отладить программу, решающую следующие задачи:

а) загрузить **изображение** с помощью функции ***imread.*** Если изображение цветное (состоит из трех компонент), преобразовать его в полутоновое с помощью функции ***rgb2gray***;

б) с помощью функции ***size*** определить пиксельный размер изображения в виде вектора *[M, N]*;

в) определить максимальное *I*max и минимальное *I*min значения яркостей пикселов изображения.

Для решения задач а), б) и в) использовать созданную в задании1 М-функцию.

5. Требования к отчёту

Отчёт должен содержать:

– титульный лист с указанием названия ВУЗа, кафедры, номера и темы лабораторной работы, а также фамилии И.О. студента, подготовившего отчёт;

– цель выполняемой работы;

– задания;

– листинги всех программ с обязательными комментариями;

– полученные на каждом этапе работы изображения;

– анализируемые в задании 3 матрицы фрагментов изображений;

– выводы по каждому выполненному заданию.

6. Контрольные вопросы

6.1. Для чего служит пакет расширения Image Processing Toolbox?

6.2. Как можно представить и описать аналоговые монохромное и цветное изображения?

6.3. Какие операции необходимо выполнить, чтобы преобразовать аналоговое изображение в цифровое?

6.4. Как можно представить и описать цифровые монохромное и цветное изображения?

6.5. Что представляет собой система координат цифрового изображения в системе MATLAB?

6.6. Понятие классов данных в MATLAB. Классы данных double, uint8 и logical.

6.7. Типы изображений в MATLAB. Полутоновые и двоичные изображения.

6.8. Чем вызвана необходимость конвертирования классов данных и типов изображений?

6.9. Как можно определить класс данных и тип изображений?

ЛАБОРАТОРНАЯ РАБОТА №4
ПРОСТРАНСТВЕННАЯ ФИЛЬТРАЦИЯ ИЗОБРАЖЕНИЙ. ПРЕОБРАЗОВАНИЕ ЯРКОСТИ И КОНТРАСТА

1. Цель работы

Изучение возможностей пакета Image Processing Toolbox. Изучение основных градационных преобразований изображений. Получение практических навыков в составлении программ.

2. Теоретический материал

Градационные преобразования Яркость и контрастность изображения являются важнейшими характеристиками изображений. *Яркость* (brightness) характеризует интенсивность излучаемого или отраженного деталями изображения света и является энергетической характеристикой. Единицей измере-ния яркости служит кд/м2. Некоторые значения яркости для ориентировки приведены ниже:

– солнце – 1,5*109 кд/м2;

– нить лампы накаливания – 5*106 кд/м2;

– пламя спички – 5*103 кд/м2;

– экран телевизора – 50...100 кд/м2.

Яркость цифрового изображения измеряется *интенсивностью* его пикселов и зависит от класса данных изображения: для изображений класса **double** она лежит в диапазоне 0...1, для изображений класса **uint8** – 0...255. Белым участкам изображения соответствует максимальная яркость, черным участкам – минимальная. *Контрастность* (contrast) в общем случае характеризует диапазон изменения яркости различных участков изображения, т.е. его динамический диапазон. Контраст является безразмерной величиной, а для его измерения существуют несколько выражений. Для оценки изображений, загруженных в среду MATLAB целесообразно использовать следующее выражение, которое дает нормированное значение контраста: $K=(B\max-B\min)/B0\max$. Здесь $B0\max$ максимально достижимая яркость пикселов в графическом окне MATLAB. При этом значения контраста лежат в диапазоне 0...1. Градационные или амплитудные преобразования изображений

связаны с преобразованием интенсивности (яркости) пикселов. Поскольку они производятся в плоскости изображений их можно отнести к простейшей обработке цифрового изображения в пространственной области, т.е. к простейшей пространственной фильтрации [1]. Процессы в пространственной области можно обозначить уравнением $g(x, y)=T[f(x, y)]$, (5.1) где $f(x, y)$ – входное изображение, $g(x, y)$ – выходное (обработанное) изображение, а T – некоторый оператор (преобразование) над f, который определен в некоторой окрестности точки с координатами (x, y). (Кроме того, оператор T может обрабатывать последовательность изображений, например, он может суммировать K входных изображений для подавления шума). В качестве окрестности точки с координатами (x, y) используется квадратная или прямоугольная область с центром в точке (x, y), которую чаще всего называют маской (рис. 5.1).

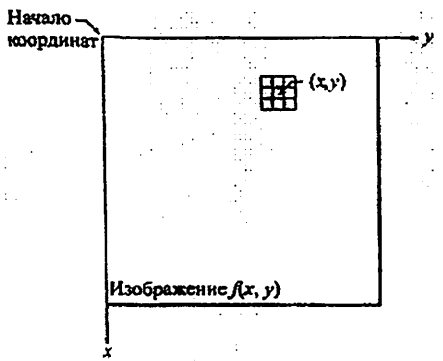

Рис. 4.1 – Окрестность (маска) 3x3 пиксела вокруг точки (x, y) изображения

Центр заданной маски перемещается от пиксела к пикселу, начиная, например, из верхнего левого угла, и на своем пути накрывает различные окрестности изображения. Преобразование T применяется в каждой точке (x, y), давая в результате обработанное значение g для данной точки. В процессе вычислений на каждом шаге используются только пикселы покрытые маской с

центром *(x, y)*. В случае градационных преобразований маска имеет размер 1x1 пиксел. При этом интенсивность обработанного пиксела *g* зависит только от интенсивности исходного пиксела, а преобразование *T* является функцией преобразования интенсивностей. В этом случае выражение (4.1) можно записать в простой форме

$$s = T(r),$$

где *s* – яркость пикселов выходного изображения *g(x, y)*, *r* – яркость пикселов входного изображения *f(x, y)*, *T* – функция преобразования интенсивностей.

Функции преобразования яркости

Функция ***imadjust*** является базовым инструментом пакета Image Processing Toolbox при преобразовании яркости изображений. Она имеет синтаксис:

g=imadjust(f, [low_in, high_in], [low_out, high_out], gamma),

где *g* – новое изображение, *f* – исходное изображение, при которых яркость в интервале [*low_in*, *high_in*] переходят в значения [*low_out*, *high_out*]. Входное изображение может быть класса **uint8**, **uint16** или **double**, а класс выходного совпадает с входным. Все яркостные параметры должны быть вещественными числами в диапазоне от 0 до 1. Если вместо интервалов [*low_in*, *high_in*] и [*low_out*, *high_out*] поставить пустой вектор ([ ]), то их значения будут взяты по умолчанию, равные [0,1]. Параметр *gamma* используется для задания формы кривой преобразования яркости, которую называют гамма-характеристикой. Возможные варианты гамма-характеристики приведены на рис. 4.2. Если параметр *gamma* опущен, то его значение устанавливается равным 1.

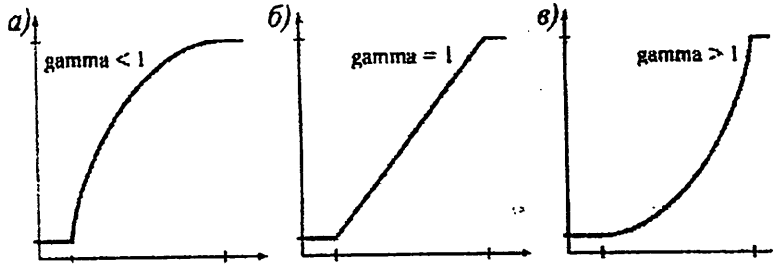

Рис. 4.2 – Возможные варианты гамма-характеристики

Негатив изображения

Процедура получения негатива изображения является весьма полезной для усиления белых и серых участков, окруженных большими, преимущественно темными, областями. Негативное изображение можно построить, используя функцию ***incomplement*** из пакета Image Processing Toolbox: ***g= incomplement (f)***, где *f* – исходное изображение. Цифровым эквивалентом получения негатива изображения является функция ***g=imadjust (f, [0, 1], [1, 0])***.

Гистограмма яркости и работа с ней Функции преобразования изображений, основанные на информации, которая извлекается из гистограмм яркости, играют ключевую роль при обработке изображений, совершаемой при решении задач улучшения изображений, их сжатия, сегментации и описания.

Гистограммой цифрового изображения с число возможных уровней яркости *L*, лежащих в диапазоне [0, G], называется дискретная функция ***h(rk)***= ***n***k, где *r*k – это k-ый уровень яркости из интервала [0, G], а ***n***k – число пикселов изображения, уровни яркости которых равен *r*k. Индексы в MATLAB начинаются с 1, а не с 0. Часто бывает удобно работать с нормированными гистограммами, которые получаются делением элементов ***h(rk)*** на общее число пикселов изображения *n*:

p(rk)=h(rk)/n = nk/n при k = 1,2,..,L.

С точки зрения теории вероятностей, число *p(rk)* – это вероятность (частота) появления (присутствия) уровня интенсивности *r*k в данном изображении.

Функция ***imhist*** служит построения гистограмм распределения яркости и используется в форме

h=imhist (f, b),

где *h* – гистограмма изображения, *f* – входное изображение, *b* – число отображаемых града-ций шкалы яркости (если не указано, то по умолчанию *b*=256).

Чтобы получить нормированную гистограмму, надо разделить полученную гистограмму на общее число пикселов изображения:

p = imhist(f, b)/numel(f).

Numel(f) дает число элементов массива *f*, то есть число пикселов изображения.

Гистограммы распределения яркостей типичного изображения естественных объектов имеют явно выраженную неравномерность, поэтому контраст такого изображения часто получается низким. ***Эквализация гистограммы*** подразумевает такое преобразование исходного изображения, при котором его гистограмма распределения яркости отвечала бы равномерному закону распределения. Иными словами, данное преобразование порождает изображение, уровни яркости которого являются равновероятными и покрывают весь интервал [0, 1]. Результатом этого процесса является большая контрастность выходного изображения.

Эквализация гистограммы реализована в пакете Image Processing Toolbox функцией ***histeq***:

g=histeq(f, nlev),

где *f* – исходное изображение, *nlev* – число уровней яркости для выходного изображения *g*. Параметр *nlev* по умолчанию имеет значение 64, поэтому для истинной реализации метода гистограммной эквализации

рекомендуется использовать *nlev* = 256. Гистограммная эквализация совершается преобразованием, которое является адаптивным в том смысле, что оно зависит от гистограммы исходного изображения. Однако если функция преобразования уже вычислена, то она не будет меняться, пока не изменится само изображение. Вычислить функцию преобразования гистограммы, построенную на основе гистограммной эквализации, можно основываясь на том, что она является кумулятивной суммой величин нормированной гистограммы:

```
%-------построение функции преобразования гистограммы----------
>>hnorm=imhist(f)./numel(f);          %нормируем гистограмму
>>sdf=cumsum(hnorm); %вычисляем функцию кумулятивного распределения
      %CDF (Cumulative Distribution Function)
>>x =linspace(0, 1, 256);   %задаем масштаб графика по оси X
>>figure; plot (x,sdf);      %строим график
>>axis([0 1 0 1]);
>>set(gca, 'xtick', 0:0.2:1)              %наносим метки на оси X
>>set(gca, 'ytick', 0:0.2:1)              %наносим метки на оси Y
>>xlabel ('Input intensity','fontsize', 9)   %надпись на оси X
>>ylabel ('Output intensity','fontsize', 9)  %надпись на оси Y
>>text(0.18, 0.5, 'Transformation function', 'fontsize', 9) %текст на графике
```

Гистограммная эквализация улучшает изображение путем расширения диапазона его уровней до более широкой шкалы яркости. Такая процедура не всегда приводит к удовлетворительному результату. Поэтому в конкретных приложениях полезно уметь задавать форму гистограммы, которую желательно иметь для обработанного изображения. Метод построения обработанного изображения с заданной гистограммой называется *гистограммной подгонкой* или гистограммной спецификацией. В пакете Image Processing Toolbox данный метод реализован в функции ***histeq*** со следующим вариантом синтаксиса:

g=histeq(f, hspec),

где f – это входное изображение, *hspec* – заданная гистограмма (вектор-строка), g – выходное изображение, гистограмма которого близка к заданной гистограмме *hspec*. Этот вектор должен состоять из целых чисел, соответствующих одинаковым разбиениям диапазона уровней.

3. Подготовка к работе

3.1. Ознакомиться с теоретическим материалом и рекомендованной литературой.

3.2. Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

Задание 1

1) Создать и отладить программу, решающую следующие задачи:

а) загрузить **изображение 1** с помощью функции *imread*. Если выяснено, что изображение цветное (состоит из трех компонент), преобразовать его в полутоновое с помощью функции *rgb2gray*;

б) с помощью функции *size* определить пиксельный размер изображения в виде вектора [M, N];

в) определить максимальное *I*max и минимальное *I*min значения яркостей пикселов изображения и вычислить контраст K;

г) определить координаты первого максимума ($r1$, $c1$) и минимума ($r2$, $c2$);

Для решения задач а), б), в) и г) создать М-функцию.

д) вывести загруженное изображение в графическое окно с помощью функции *imshow*, при этом сформировать пояснительные надписи: **Original, MxN, Imax=xxx(r1,c1), Imin=xxx(r2,c2), K=xxx**. Для того, чтобы не изменять исходное изображение при выводе на экран следует использовать функцию ***imshow*** **без параметров;**

е) используя функцию *imadjust* изменить контраст изображения, преобразовав минимальное и максимальное значения яркости до значений,

соответствующих варианту задания (таблица 5.1). Параметр *gamma* выбрать исходя из условий получения наилучшей информативности;

ж) определить максимальное *I*max и минимальное *I*min значения яркостей пикселов и кон-траст *K* обработанного изображения;

з) вывести полученное изображение в графическое окно с помощью функции ***imshow***, при этом сформировать пояснительные надписи: **Rezalt1, MxN, Imax=xxx(r1,c1), Imin=xxx(r2,c2), K=xxx;**

и) преобразовать полученное изображение в негативное;

к) определить максимальное *I*max и минимальное *I*min значения яркостей пикселов и кон-траст *K* обработанного изображения;

л) вывести полученное изображение в графическое окно с помощью функции ***imshow***, при этом сформировать пояснительные надписи: **Negative, MxN, Imax=xxx(r1,c1), Imin=xxx(r2,c2), K=xxx.**

м) сохранить полученные изображения на диск.

2) Проанализировать изображения, выведенные в графические окна, с помощью встроенного инструментария:

а) с помощью инструмента **Data Cursor** найти и нанести на изображение информацию о пикселах с максимальной и минимальной яркостью.

Задание 2

1) Создать и отладить программу, решающую следующие задачи:

а) загрузить изображение, полученное в пункте з) **задания 1;**

б) вычислить среднюю яркость изображения *I*ср;

в) вывести изображение на экран, сформировав поясняющую надпись **Original, Icp=xxx;**

г) используя функцию ***imadjust*** выполнить усиление контраста для диапазона средних значений яркости [(*I*ср–0,25*I*ср), (*I*ср+0,25*I*ср)], преобразовав его к максимальному диапазону [0, 1]. Параметр *gamma* выбрать исходя из условий получения наилучшей информативности;

д) вывести полученное изображение на экран, сформировав поясняющую надпись **Rezalt2, gamma=xx**;

е) сохранить полученное изображение на диск.

2) Проанализировать изображения, выведенные в графические окна, с помощью встроенного инструментария. Сделать выводы.

Задание 3

1) Создать и отладить программу, решающую следующие задачи:

а) загрузить **изображение 2**, указанное в таблице с вариантами заданий;

б) построить гистограмму распределения яркостей исходного изображения;

в) вывести исходное изображение и его гистограмму в одно окно экрана размером 1x2, сформировав поясняющие надписи **Original**;

г) выполнить эквализацию гистограммы исходного изображения;

д) вывести преобразованное изображение и его гистограмму в одно окно экрана размером 1x2, сформировав поясняющие надписи **Rezalt**;

е) построить и вывести в окно на экране ***характеристику преобразования гистограммы*** (последовательность команд для выполнения задачи е) оформить в виде М-функции);

ж) вывести все полученные изображения и гистограммы в одно окно, размером 2x2, сформировав поясняющие надписи;

з) сохранить все полученные изображения в файл на диске.

2) Выполнить анализ полученных изображений. Сделать и записать выводы.

Таблица 4.1

Варианты для выполнения заданий

№	Low_out	high_out	Изображение 1	Изображение 2
1	0	1	foto_1_0	foto_6_0
2	0.1	0.9	foto_1_1	foto_6_1

3	0.15	1	foto_5_1	foto_6_2
4	0.05	0.85	foto_4_0	foto_6_3
5	0.1	1	foto_4_1	foto_7_0
6	0	0.9	foto_4_2	foto_7_1
7	0	1	foto_3_0	foto_7_2
8	0.2	0.9	foto_3_1	foto_7_3
9	0.15	0.95	foto_3_2	foto_8_0
10	0.05	0.95	foto_3_3	foto_8_1

5. Требования к отчёту

Отчёт должен содержать:

– титульный лист с указанием названия ВУЗа, кафедры, номера и темы лабораторной работы, а также фамилии И.О. студента, подготовившего отчёт;

– цель выполняемой работы;

– задания;

– листинги всех программ с обязательными комментариями;

– полученные на каждом этапе работы изображения;

– выводы по каждому выполненному заданию.

6. Контрольные вопросы

6.1. Для чего служит пакет расширения Image Processing Toolbox?

6.2. Поясните необходимость и суть градационных преобразований изображений.

6.3. Дайте определение яркости и контраста изображения.

6.4. Почему градационные преобразования можно считать простейшей пространственной обработкой?

6.5. Для чего служит функция ***imadjust***? Какой синтаксис она имеет?

6.6. Как влияет на восприятие изображений гамма-характеристика?

6.7. Какими способами можно получить негатив исходного изображения? Опишите их.

6.8. Что такое гистограмма яркости? Используя какую функцию можно её получить?

6.9. Нормированная гистограмма и способы ее получения.

6.10. Для чего используется эквализация гистограммы и в чем она заключается?

6.11. Что представляет собой функция преобразования гистограммы и как можно её построить?

6.12. С помощью каких функций можно определить экстремальные значения яркости пикселов изображения и их координаты?

ЛАБОРАТОРНАЯ РАБОТА №5
ПРОСТРАНСТВЕННАЯ ФИЛЬТРАЦИЯ ИЗОБРАЖЕНИЙ. ПОДАВЛЕНИЕ ИМПУЛЬСНЫХ ШУМОВ

1. Цель работы

Изучение возможностей пакета Image Processing Toolbox. Изучение основных типов шумов на изображениях, методов их моделирования. Изучение методов фильтрации импульсных шумов. Получение практических навыков в составлении программ.

2. Теоретический материал

Восстановление – одна из задач обработки изображений. Целью восстановления является реконструкция изображения, которое ранее было искажено или испорчено процессами, информация о которых априори известна. Поэтому методы восстановления основаны на моделировании процессов

искажения и применения обратных процессов для получения исходных изображений.

Шумы и их классы

Шум является одной из основных причин приводящих к искажению изображений. Основные источники шума на цифровом изображении связаны с процессами формирования изображения в датчике и в устройстве отображения, а также с процессами связанными с преобразованием и передачей изображения по каналам передачи. Значения пространственного шума являются случайными величинами, которые, тем не менее, можно описать математически. Существуют несколько важных типов шумов, каждый из которых обладает специфическими пространственными характеристиками. Основной характеристикой шума является *функция плотности распределения вероятности* (PDF, Probability Density Function):

– Гауссов (нормальный) шум;

– шум Релея;

– шум Эрланга (гамма шум);

– экспоненциальный шум;

– равномерный шум;

– импульсный шум.

В системе MATLAB (Image Processing Toolbox) существует возможность формирования и наложения на изображение шумов. Для этого используется встроенная функция *imnoise*, которая предназначена для добавления в изображение определенного типа шума с заданными характеристиками. С помощью этой функции можно сформировать и наложить на изображение три типа шума:

1) ***gaussian*** – гауссовый белый шум (нормальный). Имеет равномерный спектр во всей полосе пространственных частот и проявляется на изображении в виде характерных посторонних вкраплений;

2) ***salt & pepper*** – импульсный шум, который проявляется в виде включенных или выключенных пикселов;

3) ***speckle*** – мультипликативный шум, который вводится в изображение в результате перемножения изображения и равномерного шума с нулевым средним значением.

Функция ***imnoise*** имеет следующий синтаксис:

g=imnoise (f, type, parameters),

где *f* –исходное изображение, *type* – тип задаваемого шума, *parameters* – аргументы, которые позволяют задать дополнительно параметры шума. Например:

g=imnoise(f, 'gaussian', m, var) добавляет к изображению *f* гауссовский белый шум со средним значением *m* и отклонением (дисперсией) *var* (по умолчанию *m*=0, *var*=0.01).

g=imnoise (f, 'salt & pepper', d), где *f* –исходное изображение, *d* – плотность шума. Добавляет импульсный шум в виде черных и белых пикселов. Плотность шума *d* приблизительно равна проценту изображения, поврежденного шумом. Число шумовых пикселов можно оценить по формуле ***d*numel(f)***. По умолчанию *d*=0.05.

g=imnoise(f, 'speckle', var) добавляет к изображению *f* мультипликативный шум по формуле *g=f+n*f*, где *n* – равномерно распределенный шум со средним значением равным 0 и отклонением (дисперсией) *var* (по умолчанию *var*=0.04).

Подавление шума методами пространственной фильтрации

Процессы при фильтрации в пространственной области можно обозначить уравнением ***g(x, y)=T[f(x, y)]***, где *f(x, y)* – входное изображение, *g(x, y)* – выходное (обработанное) изображение, а *T* – некоторый оператор (преобразование) над *f*, который определен в некоторой окрестности точки с координатами *(x, y)*. В качестве окрестности точки с координатами *(x, y)*

используется квадратная или прямоугольная область с центром в точке *(x, y)*, которую чаще всего называют маской (рис. 5.1).

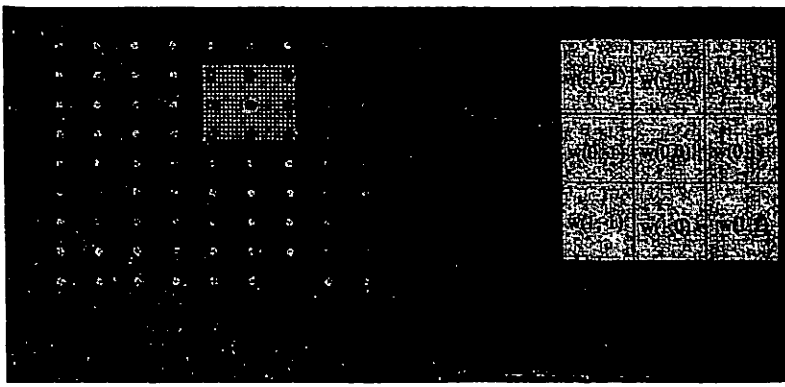

Рис. 5.1 – Принцип пространственной фильтрации

Процесс фильтрации основан на перемещении маски фильтра от точки к точке исходного изображения и состоит из последовательности следующих действий:

1) определение центральной точки с координатами *(x, y)*;

2) совершение линейной или нелинейной операции, которая использует значения пикселов, покрываемых маской в окрестности вокруг центральной точки *(x, y)*;

3) назначении результата этой операции «откликом» в этой точке;

4) повторение всего процесса для каждой точки изображения.

Отклик фильтра по маске *m* x *n* определяется выражением

$R=w_1Z_1+w_2Z_2+\ldots+w_{mn}Z_{mn}=\sum w_iZ_i$

где w_i – коэффициенты маски; Z_i –значения яркости пикселов, покрываемых маской; m, n – размеры маски.

Для подавления высокочастотных шумов, которые проявляются в виде точек с размерами соизмеримыми с размерами пикселов изображения, часто

используют однородный усредняющий фильтр, маска которого, для размера 3x3, показана на рис. 5.2.

Отклик, формируемый таким фильтром будет равен

$R=1/9 \sum Z_i$

Отсюда следует, что на каждом шаге обработки яркость центрального пиксела под маской будет заменяться средним значением яркости пикселов, покрываемых маской. Поскольку яркость пикселов, содержащих шум характерна резкими выбросами, она будет заменена на средние значения по окрестности, что приведет к снижению шумовой составляющей в изображении.

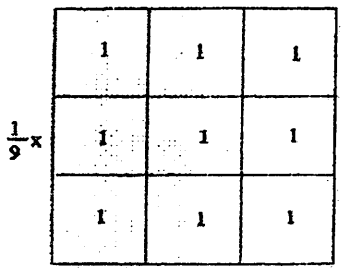

Рис. 5.2 – Маска однородного усредняющего фильтра размером 3x3

В пакете Image Processing Toolbox *линейная пространственная фильтрация* реализована функцией ***imfilter***, которая имеет следующий синтаксис:

g=imfilter (f, w, filtering_mode, boundary_options, size_options),

где *g* – результат фильтрации, *f* – входное изображение, *w* – фильтрующая маска, *filter-ing_mode* – параметр, определяющий, что совершает фильтр, корреляцию ('corr') или свертку ('conv'), *boundary_options* – опция, отвечающая за расширение границ, причем размеры рас-ширения определяются размерами фильтра, *size_options* – либо 'full', либо 'same'. Описание опций функции ***imfilter*** приводится в табл. 6.1. Чаще всего функция применяется в виде ***g=imfilter (f, w, 'replicate')***. Здесь по умолчанию используется метод свертки

изображения с фильтрующей маской *w*, а для устранения краевых эффектов используется расширение исходного изображения методом его повторения (см. табл. 5.1). Каждый элемент обработанного изображения вычисляется с использова-нием двойной точности с плавающей точкой, но в завершение работы функция ***imfilter*** конвертирует выходное изображение в класс входного. При этом некоторые вычисленные значения могут быть потеряны. Поэтому перед применением функции ***imfilter*** полезно преобразовать исходное изображение в класс **double**. Например:

Таблица 5.1

Опции функции ***imfilter***

Параметр	Опция	Описание
filtering_mode	'corr'	Фильтрация производится методом корреляции (по умолчанию).
	'conv'	Фильтрация производится методом свертки
boundary_options	'replicate'	Размер изображения увеличивается повторением величин на его боковых границах.
	'symmetric'	Размер изображения увеличивается путем зеркального отражения через границы.
	'circular'	Размер изображения увеличивается периодическим повторением двумерной функции.
	P	Границы изображения расширяются значением P (без апострофов. по умолчанию P=0).
size_options	'full'	Выход имеет те же размеры, что и

		расширенное входное изображение.
	'same'	Выход имеет те же размеры, что и вход. Это достигается с помощью ограничения перемещения центра фильтрующей маски точками, принадлежащими исходному изображению. Опция по умолчанию.

%фильтрация усредняющим фильтром

>>F=imread('1.tif'); %загрузить изображение из файла

>>F= im2double (F); %преобразовать в класс double

>>imshow(F); %вывести исходное изображение в окно

>> w = ones(3, 3)/9; %создать маску фильтра 3x3 состоящую из 1

>>G=imfilter(F,m,'replicate');%выполнить пространственную фильтрацию

>>figure; imshow(G) %вывести результат фильтрации в окно

Подавление шумов с помощью усредняющего фильтра обладает существенным недостатком, который заключается в том, что в процессе усреднения происходит размывание резких переходов яркости деталей изображения. Это приводит к снижению резкости обработанного изображения.

Медианная фильтрация

Медианный фильтр относится к классу нелинейных фильтров, основанных на порядковых статистиках (ранговых фильтров). Отклик таких фильтров определяется:

1) предварительным упорядочиванием (ранжированием) значений яркости пикселов изображения, покрываемых маской фильтра;

2) последующим выбором значения, находящегося на определенной позиции упорядоченной последовательности (т.е. имеющего определенный ранг) и назначения его откликом.

Собственно фильтрация сводится к замещению исходного значения пиксела изображения в центре маски на полученный отклик.

Медиана набора чисел – это такое число *X*, для которого половина чисел из набора меньше или равны *X*, а другая половина – больше или равны *X*. Следовательно, медиана упорядоченного набора из 9 чисел всегда равна 5-му числу, из 25 чисел равна 13-му числу и т.д. Таким образом, для выполнения медианной фильтрации следует выполнить следующую последовательность:

1) упорядочить по возрастанию значения пикселов, покрываемых маской;

2) найти значение медианы;

3) присвоить полученное значение обрабатываемому пикселу.

Медианная фильтрация весьма эффективна при удалении с изображений импульсных шумов. В пакете IPT медианный фильтр можно реализовать функцией

g = medfilt2(f, [m n], padopt),

где *f* – входное изображение; [*m n*] – размер маски фильтра; *padopt* – опции расширения изображения. По умолчанию изображение расширяется нулями.

```
%медианная фильтрация 3x3
>>f=imread('.\chess.tif');                %загрузить изображение
>>figure; imshow (f);                     %вывести изображение в окно
>>  f1=imnoise(f, 'salt & pepper');       %добавить к изображению
%импульсный шум
>> figure; imshow (f1);                   %вывести изображение в окно
>> f1=im2double(f1);                      %преобразовать в формат double
>>g1=medfilt2 (f1, [3 3]);                %выполнить фильтрацию медианным
                                          %фильтром 3x3
>>figure; imshow (g1);                    %вывести обработанное изображение
```

3. Подготовка к работе

3.1. Ознакомиться с теоретическим материалом и рекомендованной литературой.

3.2. Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

Организовать ввод данных и вычисления согласно заданиям ниже.

Задание 1

1) Создать и отладить программу, решающую следующие задачи:

а) загрузить **изображение 1** с помощью функции ***imread***. Если выяснено, что изображение цветное (состоит из трех компонент), преобразовать его в полутоновое с помощью функции ***rgb2gray***;

б) вывести загруженное изображение в графическое окно с помощью функции ***imshow***, при этом сформировать пояснительные надписи: **Original.** Для того, чтобы не изменять исходное изображение при выводе на экран следует использовать функцию ***imshow*** без параметров;

в) используя функцию ***imnoise*** добавить к изображению нормальный гауссовский шум с параметрами , взятыми по умолчанию;

г) вывести полученное изображение в графическое окно с помощью функции ***imshow***, при этом сформировать пояснительные надписи: **Original+Gaussian, var=xxx;**

д) повторить пункты в) и г) задания для импульсного и мультипликативного шума.

2) Сравнить результаты влияния шума на исходное изображение. Сделать выводы.

3) Повторить все пункты задания для **изображения 2.**

Задание 2

1) Создать и отладить программу, решающую следующие задачи:

а) загрузить **изображение 1** с помощью функции ***imread***. Если выяснено, что изображение цветное (состоит из трех компонент), преобразовать его в полутоновое с помощью функции ***rgb2gray***;

б) вывести загруженное изображение в графическое окно с помощью функции ***imshow***, при этом сформировать пояснительные надписи: **Original**. Для того, чтобы не изменять исходное изображение при выводе на экран следует использовать функцию ***imshow*** без параметров;

в) используя функцию ***imnoise*** добавить к изображению высокочастотный шум типа «соль и перец» с плотностью, указанной в столбце **d1,2** таблицы с вариантами заданий;

г) вывести полученное изображение в графическое окно с помощью функции ***imshow***, при этом сформировать пояснительные надписи: **Original+Noise d=xxx**;

д) сформировать усредняющий фильтр размером m x n = 3x3 и выполнить фильтрацию с целью устранения высокочастотного шума;

е) вывести полученное изображение в графическое окно с помощью функции *imshow*, при этом сформировать пояснительные надписи: **Rezalt , average mxn**;

ж) повторить пункты д) и е) задания для усредняющего фильтра размерами m x n = 5x5 и 7x7;

з) сохранить полученные изображения в файл.

2) Проанализировать изображения, выведенные в графические окна, сделать выводы.

Задание 3

1) Создать и отладить программу, решающую следующие задачи:

а) загрузить **изображение 1** с помощью функции ***imread***. Если выяснено, что изображение цветное (состоит из трех компонент), преобразовать его в полутоновое с помощью функции ***rgb2gray***;

б) вывести загруженное изображение в графическое окно с помощью функции ***imshow***, при этом сформировать пояснительные надписи: **Original.** Для того, чтобы не изменять исходное изображение при выводе на экран следует использовать функцию ***imshow*** без пара-метров;

в) используя функцию ***imnoise*** добавить к изображению высокочастотный шум типа «соль и перец» с плотностью, указанной в столбце d1,2 таблицы с вариантами заданий;

г) вывести полученное изображение в графическое окно с помощью функции ***imshow***, при этом сформировать пояснительные надписи: **Original+Noise d=xxx;**

д) сформировать медианный фильтр с маской размерами m x n = 3x3 и выполнить нелинейную фильтрацию с целью устранения высокочастотного шума;

е) вывести полученное изображение в графическое окно с помощью функции ***imshow***, при этом сформировать пояснительные надписи: **Rezalt , median mxn.** Оценить результат;

ж) повторить пункты д) и е) задания для медианного фильтра размерами ***mxn***= 5x5 и 7x7;

з) сохранить полученные изображения в файл.

2) Проанализировать изображения, выведенные в графические окна, сделать выводы.

Задание 4

Повторить все пункты задания 1 для **изображения 2.** Значения плотности шума взять из столбца таблицы d3,4.

Задание 5

Повторить все пункты задания 2 для **изображения 2.** Значения плотности шума взять из столбца таблицы d3,4.

Таблица 5.2

Варианты для выполнения заданий

№ варианта	Плотность шума d1,2	Плотность шума d3,4	Изображение1	Изображение2
1	0.05	0.35	foto_6_1	foto_6_0
2	0.075	0.3	foto_6_2	foto_6_9
3	0.085	0.25	foto_6_3	foto_6_8
4	0.095	0.2	foto_6_4	foto_6_7
5	0.1	0.35	foto_6_5	foto_6_6
6	0.15	0.3	foto_6_6	foto_6_5
7	0.2	0.05	foto_6_7	foto_6_4
8	0.25	0.05	foto_6_8	foto_6_3
9	0.3	0.075	foto_6_9	foto_6_2
10	0.35	0.085	foto_6_0	foto_6_1

5. Требования к отчёту

Отчет должен содержать:

– титульный лист с указанием названия ВУЗа, кафедры, номера и темы лабораторной работы, а также фамилии И.О. студента, подготовившего отчет;

– цель выполняемой работы;

– задания;

– листинги всех программ с обязательными комментариями;

– полученные на каждом этапе работы изображения;

– выводы по каждому выполненному заданию.

6. Контрольные вопросы

6.1. Что понимается под термином восстановление изображений ?

6.2. Какая функция IPT используется для добавления шума к изображению? Что означают ее параметры?

6.3. Какие три основных типа шума можно формировать средствами MATLAB?

6.4. Расскажите о функции добавления гауссового шума к изображению.

6.5. Расскажите о функции добавления импульсного шума к изображению.

6.6. Расскажите о функции добавления мультипликативного шума к изображению.

6.7. Расскажите о принципах пространственной фильтрации.

6.8. Какие характеристики может иметь фильтрующая маска?

6.9. Какие методы используют для подавления импульсных шумов?

6.10. Расскажите о функции, реализующей линейную пространственную фильтрацию.

6.11. Какие опции функции *imfilter* существуют? За что они отвечают?

6.12. Поясните принцип работы однородного усредняющего фильтра.

6.13. Расскажите о функции, реализующей однородный усредняющий фильтр.

6.14. Поясните принцип работы медианного фильтра.

6.15. Расскажите о функции, реализующей медианную фильтрацию.

ЛАБОРАТОРНАЯ РАБОТА №6
РАБОТА С ФАЙЛАМИ ИЗОБРАЖЕНИЙ

1. Цель работы

Целью работы является изучение пакета прикладных программ Image Processing Toolbox, входящего в состав среды имитационного моделирования Matlab. Пакет включает в себя более 100 функций, реализующих наиболее часто встречающиеся методы обработки изображений.

2. Теоретический материал

Одной из самых частых операций по работе с изображениями является их считывание из графических файлов с целью загрузки в рабочее пространство MATLAB. Эта операция реализуется функцией ***imread***:

imread('filename').

Здесь *filename* – это строка символов, образующих полное имя загружаемого файла изображения (включая любое расширение). Для случая, когда графический файл включен в приложение Image Processing Toolbox, достаточно указать только имя файла и его расширение, например:

>> A=imread('image.bmp');

При этом матричной переменной *A* присваивается значения матрицы, соответствующей изображению, хранящемуся в файле с именем *image.bmp*. Точка с запятой в конце командной строки запрещает (подавляет) вывод результата. Для данной команды запрещается вывод на экран значений массива *A*. В случае, когда изображение расположено в конкретной папке, необходимо в явном виде указать полный путь к этой директории, например:

>> A=imread('D:\Image\image.jpeg').

Для вывода на дисплей изображения, находящегося в рабочем пространстве, используется функция ***imshow***, которая имеет следующий синтаксис:

imshow (A, G),

где *A* – матрица выводимого изображения, *G* – число градаций яркости, используемое для отображения изображения *A*. Если *G* отсутствует, по умолчанию используется 256 градаций яркости.

Команда

imshow(f, [low high])

позволяет вывести изображение в котором все пикселы с интенсивностью не больше числа *low* отображаются черными, а пикселы с интенсивностью не меньше числа *high* отображаются белыми. Если значения *[low high]* не

задаются, а квадратные скобки остаются пустыми, то пикселы с минимальной интенсивностью отображаются черными, а пикселы с максимальной интенсивностью – белыми, т.е. вывод в графическое окно производится с максимальным контрастом. Следующая последовательность команд производит загрузку изображения **1.tif** из папки на диске **D** в рабочее пространство и выводит графическое окно с изображением на экран:

>> A=imread('D:\Image\1.tif');

imshow(A)

Результат выполнения команд изображен на рис. 6.2.

Рис. 6.2 – Изображение, выведенное на экран

Для того, чтобы вывести несколько изображений в отдельных графических окнах, необходимо выполнить последовательность команд:

>> imshow (A);

>> figure, imshow (B); title(['Image B']);

>> figure, imshow (C); title(['Image C']);

где *A* – матрица первого изображения, *B* – матрица второго изображения, *C* – матрица третьего изображения. Команда ***title(['string1', 'string2', '...'])***

позволяет добавить надписи в выводимое окно, которые необходимо указывать в апострофах. Если в надпись нужно ввести значения переменных, то это производится следующей конструкцией: ***title(['string1', 'string2', '...', num2str(имя перем.1), num2str(имя прем.2), ...])***.

Для сохранения изображения в файл используется функция ***imwrite***, имеющая формат:

imwrite (A, 'filename'),

где *A* – матрица изображения, *'filename'* – имя сохраняемого файла с указанием расширения, поддерживаемого системой MATLAB. Если *'filename'* не содержит информацию о пути к папке, то сохранение производится в текущую папку.

Зачастую возникает необходимость сохранения изображения на диске точно в том виде, в каком оно отображено в графическом окне на экране. Содержимое окна изображений можно экспортировать на диск с помощью команды ***Export*** в ниспадающем меню File в окне изображений (см. рис. 6.2).

Для того, чтобы получить *информацию о размере изображений*, необходимо использовать функцию ***size(A)***, которая возвращает размер изображения *A* в виде вектора из двух элементов *M* и *N*.

Пример: *>> size (A)*

ans =

500 500

Для того, чтобы получаемые значения размеров использовать для дальнейших вычислений следует применять функцию ***size*** в следующем виде: *>> [M N] = size (A);*

Получить информацию о массиве можно функцией ***whos A***, которая была рассмотрена выше.

Для более подробной информации об изображении можно использовать функцию ***im-ageinfo('filename')***, которая выводит метаданные об указанном файле в виде таблицы.

Пример:

>> imageinfo('D:\Image\1.tif')

Результат выполнения команды изображен на рис. 6.3.

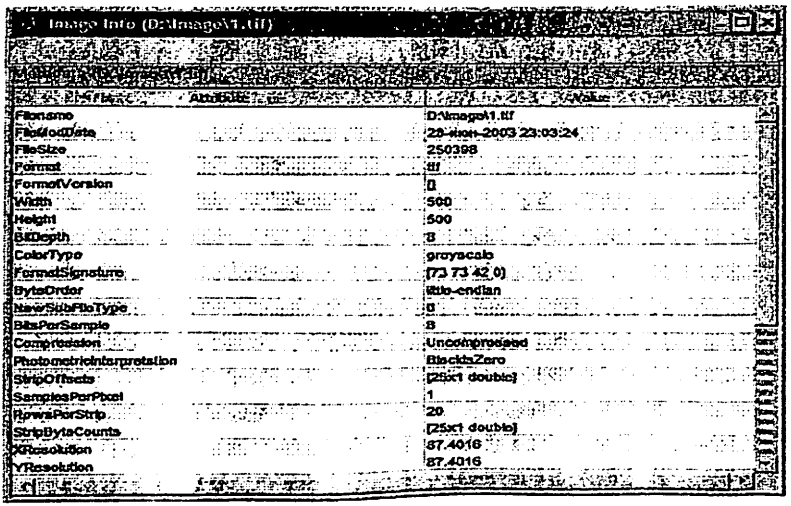

Рис. 6.3 – Подробная информация об изображении

Аналогичную информацию о файле изображения можно получить используя функцию *imfinfo*, которая имеет вид

imfinfo ('filename'),

где *filename* – это полное имя файла изображения, хранящегося на диске.

Например,

>> imfinfo ('bubbles25.jpg')

Информационные поля, которые выводятся на экран функцией ***imfinfo***, можно ввести в т.н. структурные переменные, которые затем можно использовать в последующих вычислениях. Беря в качестве примера предыдущее изображение и присваивая имя *K* структурной переменной, можно дать команду

>> K = imfinfo('bubbles25.jpg');

для сохранения в переменной *K* всей информации, генерируемой командой ***imfinfo***. Информация, полученная из функции ***imfinfo*** сохраняется в полях структурной переменной *K*, которые отделяются от нее точкой. Например, высота и ширина изображения теперь хранятся в структурных полях *K.Height* и *K.Width*, битовая глубина в поле *K.BitDepth*, а размер файла изображения в поле *K.FileSize*.

М-функции

В отличии от файлов-сценариев (скриптов), которым невозможно передать входные параметры, *М-функции* имеют *входные параметры* (*аргументы*) и допускают наличие возвращаемых *выходных параметров*. Это позволяет оформлять независимые и изолированные фрагменты программы, решающие некоторые фиксированные задачи, в виде М-функций, которые можно неоднократно использовать с помощью вызовов. Аналогом М-функций является подпрограмма или процедура, используемая при модульном программировании.

М-функции, создаваемые в М-файлах, состоят из компонент:

1) заголовок функции;

2) основной комментарий;

3) дополнительный комментарий (текст справки);

4) тело функции.

М-функция создается по следующим правилам:

– *заголовок* является первой строкой и имеет следующий вид:

function[выходные_параметры] = f_name(входные_параметры);

– *имя функции f_name* может быть произвольным, но слово ***function*** должно быть обязательно. Если определяемая функция не имеет выходных параметров, то используется одно слово ***function*** без квадратных скобок и знака равенства. Пробелы в имени функции не допускаются;

– *основной комментарий* это текстовый блок, каждая строка которого начинается с %. Между этим блоком и заголовком функции не должно быть пустых строк;

– *дополнительный комментарий* – текстовый блок, который размещается сразу после основного. Он служит для отображения комментариев и онлайн справки по данной функции;

– *тело функции* состоит из выполняемого кода MATLAB, который совершает действия или вычисления и присваивает результаты выходным аргументам;

– функция возвращает свое значение и может использоваться в виде ***f_name (входные_параметры)*** в математических выражениях;

– все переменные, имеющиеся в теле М-функции, являются *локальными*, т. е. действуют только в пределах тела функции;

– М-функция является самостоятельным программным модулем, который общается с другими модулями через свои входные и выходные параметры;

– если выходных параметров больше одного, то они указываются в квадратных скобках после слова ***function***;

– набранный в редакторе текст М-функции необходимо сохранить в файл, причем *имя файла* должно совпадать с *f_name*, указанным в заголовке.

Для примера создадим М-функцию, которая загружает изображение из файла и вычисляет некоторые его параметры:

```
function [M, N, Imax, Imin]=begin_1(f0)
 %функция begin_1 загружает изображение, преобразует его в
%полутоновое, вычисляет размер, минимальное и максимальное
 % значения яркости
 %f0 -файл входного изображения
%M, N – размер загруженного изображения
%Imax, Imin – максимальное и минимальное
% значения яркости пикселов
```

>>f=imread (f0); %загрузить изображение

>>f=rgb2gray(f); %преобразовать в полутоновое

>>whos f; %получить сведения об изображении

>> f=im2double (f); %преобразовать в класс double

>>[M,N]= size(f); %определить размер

>>Imax=max(f(:)); %определить максимум

>>Imin=min(f(:)); %определить минимум

М-функцию можно вызвать из командной строки системы MATLAB или из других М-файлов, при этом указав все необходимые атрибуты – *входные аргументы в круглых скобках, выходные аргументы в квадратных скобках.* Для рассмотренного примера вызов будет таким:

>>[M, N, Imax, Imin]=begin_1('foto_8_1.tif')

3. Подготовка к работе

3.1. Ознакомиться с теоретическим материалом и рекомендованной литературой.

3.2. Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

Организовать ввод данных и вычисления согласно заданиям ниже. Исходные изображения для выполнения работы хранятся в папке Images_6 методических указаний.

Задание 1

1) Загрузить в рабочее пространство MATLAB **изображение 1 из файла**, указанного в таблице с вариантами заданий.

2) Используя функцию ***whos*** определить тип и класс загруженного изображения. Записать результат.

3) Создать и отладить программу, решающую следующие задачи:

а) загрузить изображение с помощью функции ***imread***. Если было выяснено, что изображение цветное (состоит из трех компонент), преобразовать его в полутоновое с помощью функции ***rgb2gray***;

б) с помощью функции *size* определить пиксельный размер изображения в виде вектора *[M, N]*;

в) определить максимальное *I*max и минимальное *I*min значения яркостей пикселов изображения;

г) вывести загруженное изображение в графическое окно с помощью функции ***imshow***, при этом сформировать пояснительные надписи: **Original, MxN, Imax=xxx, Imin=xxx**. Для того, чтобы не изменять исходное изображение при выводе на экран следует использовать функцию ***imshow*** без параметров.

д) записать изображение на диск в рабочую папку, используя функцию ***imwrite (f, 'filemame')***, присвоив ему имя **zad4_1** сохранив прежнее расширение.

5. Требования к отчёту

Отчёт должен содержать:

– титульный лист с указанием названия ВУЗа, кафедры, номера и темы лабораторной работы, а также фамилии И.О. студента, подготовившего отчёт;

– цель выполняемой работы;

– задания;

– листинги всех программ с обязательными комментариями;

– полученные на каждом этапе работы изображения;

– анализируемые в задании 3 матрицы фрагментов изображений;

– выводы по каждому выполненному заданию.

6. Контрольные вопросы

6.1. Для чего служит пакет расширения Image Processing Toolbox?

6.2. Что представляет собой система координат цифрового изображения в системе MATLAB?

6.3. Понятие классов данных в MATLAB. Классы данных double, uint8 и logical.

6.4. Типы изображений в MATLAB. Полутоновые и двоичные изображения.

6.5. Чем вызвана необходимость конвертирования классов данных и типов изображений?

6.6. Как можно определить класс данных и тип изображений?

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Указ Президента Республики Узбекистан от 7 февраля 2017 года №УП-4947 «О стратегии действий по дальнейшему развитию Республики Узбекистан».

2. Matlab и Simulink – сообщество пользователей, материалы, книги, форум [Электронный ресурс] – Режим доступа: http://matlab.exponenta.ru, свободный.

3. Дьяконов, В. П. MATLAB. Полный самоучитель / В. П. Дьяконов. – М.: ДМК Пресс, 2012. – 768 с.

4. Дьяконов, В. П. MATLAB 6.5 SP1/7/7 SP1 + Simulink 5/6. Работа с изображениями и видеопотоками / В. П. Дьяконов. – М.: СОЛОН-Пресс, 2005. – 522 с.

5. Солонина, А. И. Цифровая обработка сигналов. Моделирование в MATLAB [Текст] / А. И. Солонина, С. М. Арбузов. – СПб.: БХВ-Петербург, 2008. – 816 с.

6. Mirziyoyev, Sh.M. Buyuk kelajagimizni mard va olijanob xalqimiz bilan birga quramiz. 2017.

7. Mirziyoyev, Sh.M. Qonun ustuvorligi va inson manfaatlarini ta'minlash – yurt taraqqiyoti va xalq farovonligining garovi. 2017.

8. Mirziyoyev, Sh.M. Erkin va farovon, demokratik O'zbekiston davlatini birgalikda barpo etamiz. 2017

9. Мирзиёв, Ш.М. Танқидий таҳлил, қатъий тартиб-интизом ва шахсий жавобгарлик – ҳар бир раҳбар фаолиятининг кундалик қоидаси бўлиши керак. Ўзбекистон Республикаси Вазирлар Маҳкамасининг 2016 йил якунлари ва 2017 йил истиқболларига бағишланган мажлисидаги Ўзбекистон Республикаси Президентининг нутки.// Халқ сўзи газетаси. 2017 йил 16 январь, №11.

СОДЕРЖАНИЕ

Формат 60x84 1/16. Печ.лист 5,25.
Заказ № 392. Тираж 50.
Отпечатано в «Редакционно издательском»
отделе при ТУИТ.
Ташкент ул. Амир Темур, 108.

МЕТОДИЧЕСКОЕ ПОСОБИЕ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ПРЕДМЕТУ «ОБРАБОТКА ЦИФРОВОЙ ИНФОРМАЦИИ»

5350200 - Televizion texnologiyalar (Audiovizual texnologiyalar) mutaxassisligi talabalari uchun uslubiy qo'llanma

AVT kafedrasining 2018 yil 25 sentyabr (5-sonli bayonnoma) majlisida ko'rib chiqildi va chop etishga tavsiyalandi

TT fakultetining ilmiy-uslubiy Kengashida ko'rib chiqildi va chop etishga tavsiyalandi
2018 yil 25-sentyabr 2-sonli bayonnoma

TATU ilmiy-uslubiy Kengashida ko'rib chiqildi va chop etishga tavsiyalandi
2018 yil 23-oktyabrdagi 4(116) –sonli bayonnoma

Muallif: Sh.T.Kasimova,

Taqrizchilar: Q. Raxmanov
E. Nazirova

Mas'ul muharrir: S.Beknazarova
Musahhih: N.X.Raximova