

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**

—◆—  
**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ГОРНЫЙ  
УНИВЕРСИТЕТ**  
—◆—

Кафедра физико-технического контроля процессов  
горного производства

*Утверждено на заседании Совета Учебно-методического объединения вузов Российской Федерации по образованию в области горного дела 27.01.2011, протокол № 1*

А. С. Вознесенский

## **КОМПЬЮТЕРНЫЕ МЕТОДЫ В НАУЧНЫХ ИССЛЕДОВАНИЯХ**

Часть 1. Ознакомление с системой MATLAB и ее использование  
при исследовании геосред

*Допущено Учебно-методическим объединением вузов Российской Федерации по образованию в области горного дела в качестве учебника для студентов вузов, обучающихся по специальности «Физические процессы горного или нефтегазового производства» направления подготовки «Горное дело»*

Москва 2011

**ВОЗНЕСЕНСКИЙ А. С.**

Компьютерные методы в научных исследованиях. Часть 1. Ознакомление с системой MATLAB и ее использование в автоматизированном эксперименте: Учебник для студентов специальности 130401 «Физические процессы горного или нефтегазового производства».- М.:МГГУ, 2011, 107 с.:ил.

Предназначен для студентов специальности 130401 «Физические процессы горного или нефтегазового производства». Часть 1 содержит общую информацию по системе MATLAB, а также начальные сведения, необходимые для обработки изображений и автоматизации физического эксперимента с использованием этой системы, что необходимо современному исследователю. Может представлять интерес для студентов других специальностей, а также для научных и инженерных работников, чья деятельность связана с компьютерным моделированием.

А. С. Вознесенский – д-р техн. наук, проф. кафедры «Физико-технический контроль процессов горного производства» Московского государственного горного университета.

Рецензенты:

- д-р техн. наук, проф. Парамонов А. А. [зав. кафедрой «Радиоприемные устройства» Московского государственного института радиотехники, электроники и автоматики (Технического университета)]
- д-р техн. наук, проф. Захаров В. Н. [зав. лаб. «Информационные системы геотехнологий» Института проблем комплексного освоения недр РАН (УРАН ИПКОН РАН)]

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	5
1. ОСНОВНЫЕ ПОНЯТИЯ ИНФОРМАЦИОННЫХ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ В НАУЧНЫХ ИССЛЕДОВАНИЯХ .....	7
КОНТРОЛЬНЫЕ ВОПРОСЫ К ГЛАВЕ 1 .....	9
2. ЗНАКОМСТВО С СИСТЕМОЙ MATLAB .....	10
2.1. Общие сведения .....	10
2.2. Интерфейс MATLAB .....	11
2.3. Типы данных системы MATLAB .....	13
2.4. Простейшие приемы работы с векторами и матрицами .....	19
2.5. Операторы и функции .....	22
2.6. Работа со справочной системой MATLAB.....	23
2.7. Работа с файлами данных.....	26
КОНТРОЛЬНЫЕ ВОПРОСЫ К ГЛАВЕ 2 .....	27
3. ГРАФИКА И ГРАФИЧЕСКИЕ ФОРМАТЫ.....	28
3.1. Основные понятия.....	28
3.2. Параметры графических форматов .....	29
3.3. Два режима представления графической информации.....	30
3.4. Графические редакторы .....	44
3.5. Графика в системе MATLAB .....	44
3.5.1. Создание графиков .....	44
3.5.2 Подграфики .....	48
3.5.3 Управление осями.....	49
3.5.4 Подписи к осям и заголовки .....	50
3.5.5 Функции mesh и surface.....	51
3.5.6. Визуализация функций двух переменных.....	51
3.5.7. Печать графики .....	52
КОНТРОЛЬНЫЕ ВОПРОСЫ К ГЛАВЕ 3 .....	54
4. ПОЛУЧЕНИЕ И ОБРАБОТКА ИЗОБРАЖЕНИЙ.....	55
4.1. Получение изображений .....	55
4.2. Виды обработки изображений .....	56
4.3. Первичные преобразования изображений.....	57
4.4. Определение содержания минералов в горной породе.....	62
4.5. Определение размеров минеральных агрегатов .....	68
КОНТРОЛЬНЫЕ ВОПРОСЫ К ГЛАВЕ 4 .....	76
5. АВТОМАТИЗИРОВАННЫЙ СБОР ДАННЫХ В ФИЗИЧЕСКОМ ЭКСПЕРИМЕНТЕ .....	77
5.1. Принципы компьютерного сбора данных в физическом эксперименте ..	77
5.2. Использование звуковой карты для ввода и записи аналоговых сигналов в компьютере с использованием системы MATLAB.....	81
5.2.1. Предварительные сведения о звуковой карте.....	81
5.2.2. Программирование модулей ввода/вывода.....	82
5.2.3. Непрерывная регистрация данных.....	92

5.2.4. Регистрация сигнала и вывод спектра .....	98
5.2.5. Регистрация сигналов с запуском по превышению порога срабатывания .....	100
5.3. Компиляция программ.....	103
КОНТРОЛЬНЫЕ ВОПРОСЫ К ГЛАВЕ 5 .....	104
ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ.....	105
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	107

## ВВЕДЕНИЕ

Без научных исследований сегодня невозможно представить ни одну из сфер человеческой деятельности. Первоначально исследования в области наук о Земле осуществлялись чисто умозрительно, что требовало от исследователя большого опыта и исключительных мыслительных способностей. Накопление знаний и усложнение исследований диктовали необходимость появления различных приборов, инструментов и приспособлений, позволяющих производить такие исследования глубже, тщательней и в больших объемах, чем раньше.

Объем, интенсивность и достоверность научных исследований резко увеличились в связи с появлением компьютеров. В отечественной литературе этот инструмент коротко обозначался аббревиатурой ЭВМ, что означает «электронная вычислительная машина». Практика показала целесообразность их построения по цифровому принципу.

С момента появления первых цифровых электронных вычислительных машин (ЭЦВМ) они использовались именно для вычислений. Один из основателей отечественной вычислительной техники в нашей стране С. А. Лебедев производил расчеты процессов в проводных сетях при передаче электроэнергии на расстояние, что привело его к мысли создания ЭЦВМ. Первый цифровой электронный компьютер, созданный Д. Мочли и П. Эккертом в 1946 году в США, использовался для баллистических расчетов, производимых при стрельбах. Можно сказать, что это было, по существу, компьютерное моделирование реальных физических процессов. Но развитие компьютеров позволило им выполнять и другие задачи, что значительно расширило сферу их применения.

Современные научные исследования объектов и явлений физического происхождения, с которыми приходится иметь дело в геофизике, горном производстве и других сферах науки и техники, невозможны без использования компьютеров благодаря их возможностям:

- воспринимать с клавиатуры и сохранять в памяти различные алфавитные, цифровые, символьные данные, а также осуществлять их администрирование;
- производить численные и логические вычисления по программам, которые можно легко менять в соответствии с требуемыми алгоритмами;
- вводить, выводить с помощью аналого-цифрового и цифроаналогового преобразователей, а также записывать в память и извлекать из нее сигналы в аналоговой форме, например, сигналы с преобразователя перемещений или датчика-акселерометра;
- выводить на экран в алфавитно-цифровой и графической формах различную информацию о результатах расчетов, измерений или другую, хранящуюся в памяти.

Это дает в руки исследователя-экспериментатора следующие инструментальные возможности:

- автоматический сбор и регистрацию данных об объекте исследования, а также различных сигналов, характеризующих его поведение и взаимодействие с другими объектами;
- первичное преобразование, обработку, систематизацию собранных и зарегистрированных данных, проведение расчетов на их основе;
- составление математических моделей, описывающих объект и его взаимодействие с другими объектами;
- изучение на этих моделях особенностей объекта, интересующих исследователя, и установление новых закономерностей.

Как следует из перечисленного, для успешного использования компьютеров в научных исследованиях необходимо хотя бы в общих чертах представлять, как они функционируют, уметь программировать их работу, знать, что нужно сделать, чтобы записать различные сигналы в ходе физического эксперимента, как их обработать. Кроме того, экспериментатор должен владеть набором определенных компьютерных методов, уметь составлять математические модели и проводить компьютерный вычислительный эксперимент. Каждый эксперимент должен сопровождаться визуализацией результатов, а также логичным их изложением, доступным для ясного понимания другими учеными. Освоению этих знаний и навыков посвящено настоящее издание. Его изучение предполагает выполнение лабораторно-практических заданий, позволяющих выработать и закрепить общие навыки научных исследований в области горного и нефтегазового дела с применением компьютеров.

Настоящий курс условно можно разбить на две части.

В первой из них осуществляется знакомство с системой MATLAB, используемой сейчас во многих университетах и научных лабораториях мира.

Вторая часть посвящена изучению программного продукта Comsol Multiphysics, предназначенного для моделирования различных физических процессов на основе решения дифференциальных уравнений, отличительной особенностью которого является возможность построения имитационных компьютерных моделей с использованием нескольких физических законов.

Данное учебное пособие подготовлено в результате чтения автором лекций по дисциплине «Компьютерные методы в научных исследованиях», являющейся одним из звеньев в подготовке инженеров специальности «Физические процессы горного или нефтегазового производства» в области информационных компьютерных технологий. При освоении этого курса считается, что студенты знакомы с дисциплинами «Информатика», а также «Компьютерные методы в инженерных расчетах» или аналогичными по содержанию.

# 1. ОСНОВНЫЕ ПОНЯТИЯ ИНФОРМАЦИОННЫХ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ В НАУЧНЫХ ИССЛЕДОВАНИЯХ

Научные исследования могут носить теоретический и экспериментальный характер.

*Научное исследование* - процесс изучения, эксперимента, разработки и проверки теории, связанный с получением научных знаний. Различают фундаментальные и прикладные научные исследования.

*Фундаментальные научные исследования* - экспериментальная или теоретическая деятельность, направленная на получение новых знаний об основных закономерностях строения, функционирования и развития человека, общества, окружающей природной среды.

*Прикладные научные исследования* - исследования, направленные преимущественно на применение новых знаний для достижения практических целей и решения конкретных задач.

Научные исследования часто осуществляются с помощью моделирования.

*Моделирование* – метод научного познания, основанный на изучении реальных объектов посредством изучения моделей этих объектов, т. е. посредством изучения более доступных для исследования и (или) вмешательства объектов-заместителей естественного или искусственного происхождения, обладающих свойствами реальных объектов.

*Модель* - создаваемое человеком подобие изучаемых объектов: макеты, изображения, схемы, словесные описания, карты, математические формулы, компьютерные программы и т. д.

Модели всегда проще реальных объектов, но они позволяют выделить главное, не отвлекаясь на детали. Слово «модель» происходит от латинского *modus* (копия, образ, очертание). Моделирование - это замещение одного объекта другим. Замещаемый объект называется оригиналом или объектом моделирования, а замещающий - моделью. Другими словами, модель - это объект-заменитель объекта-оригинала, обеспечивающий изучение некоторых свойств оригинала. В интересующих нас областях естествознания и техники различают физические, математические, компьютерные модели.

*Математическая модель* - модель объекта, процесса или явления, представляющая собой математические закономерности, с помощью которых описаны основные характеристики моделируемого объекта, процесса или явления.

*Физические модели* - модели, создаваемые путем замены объектов моделирующими устройствами, которые имитируют определенные характеристики либо свойства этих объектов. При этом моделирующее устройство имеет ту же качественную природу, что и моделируемый объект.

Физические модели используют эффект масштаба в случае возможности пропорционального применения всего комплекса изучаемых свойств.

В последние годы широкое распространение получило компьютерное моделирование различных объектов и процессов, позволяющее получить но-

вое знание с наименьшими затратами по сравнению с физическим моделированием. Однако результаты такого моделирования должны проверяться, либо путем наблюдений за реальными объектами, либо с помощью все того же физического моделирования. В то же время, предварительные результаты, полученные с помощью компьютера, позволят наиболее рационально провести натурные исследования.

Под *компьютерной моделью* понимают:

- условный образ объекта или некоторой системы, описанный с помощью взаимосвязанных компьютерных таблиц, блок-схем, диаграмм, графиков, рисунков, анимационных фрагментов, гипертекстов и т. д. и отображающий структуру и взаимосвязи между элементами объекта – структурно-функциональная модель;

- отдельную программу, совокупность программ, программный комплекс, позволяющие с помощью последовательности вычислений и графического отображения их результатов воспроизводить (имитировать) процессы функционирования объекта при условии воздействия на него различных (включая случайные) факторов – имитационные модели.

*Современный компьютер* - это программируемое электронное устройство, способное обрабатывать числовые и нечисловые данные и производить вычисления, а также выполнять другие задачи манипулирования символами. Сюда же нужно добавить коммуникационные возможности компьютеров, а именно - обмен данными с другими компьютерами, с преобразователями различных физических величин и в различных формах – визуальной, звуковой и т. д., а также с человеком.

Компьютерное моделирование физических процессов, адекватно отражающее действительность, может базироваться только на результатах наблюдений, проводимых в реальной жизни. Такие наблюдения сейчас проводятся тоже с использованием компьютеров. В настоящее время можно говорить о целой системе компьютерных методов, представляющих собой широкий набор инструментов исследователя, которые позволяют ему осуществлять сбор, накопление, обработку, визуализацию данных об объектах исследования, а также моделирование этих объектов. Диапазон использования таких методов весьма широк – от простейших вычислений до виртуозного владения ими в сложнейших экспериментах, включающих высокоинтеллектуальную обработку зарегистрированных в ходе эксперимента данных и моделирование сверхсложных процессов и явлений, что было немыслимо всего несколько десятилетий назад.

Эксперименты и моделирование с использованием компьютеров могут осуществляться на базе многочисленных программ, существующих в настоящее время. Как показала практика последних лет, для целей обучения и практического использования наиболее целесообразно использование программной системы MATLAB, получившей широкое распространение в университетах и научных лабораториях мира. На ее базе создана также система мультифизического моделирования с использованием метода конечных элементов COMSOL Multiphysics. Эти две системы позволяют производить вычисления и модели-



рование в связке друг с другом, что позволяет решать широкий круг задач и дает в руки исследователя мощный инструмент. Тем самым обусловлено направление, принятое в изложении материалов последующих глав.

## **КОНТРОЛЬНЫЕ ВОПРОСЫ К ГЛАВЕ 1**

1. Что такое научное исследование?
2. В чем различие фундаментальных и прикладных исследований, как они связаны между собой?
3. Что такое модель, моделирование? В чем особенность моделирования физических процессов горного производства?
4. Дайте определение математической и физической моделей, как они могут быть связаны друг с другом?
5. Что такое компьютерная модель физического объекта? Как компьютерные модели взаимодействуют с математическими и физическими моделями?
6. Какие виды компьютерных моделей используются на практике?

## 2. ЗНАКОМСТВО С СИСТЕМОЙ MATLAB

### 2.1. Общие сведения

Система MATLAB (от Matrix Laboratory – матричная лаборатория) известна во многих университетах мира. Широкому распространению системы способствует ее язык, очень похожий по своей структуре на языки Basic и Fortran, а также большое количество приложений, так называемых «ящиков с инструментами» или инструментариями (Toolboxes), количество которых все время увеличивается. Это дает большие возможности при обучении студентов, особенно в части получения ими практических навыков при освоении тех или иных разделов курсов по обработке данных и моделировании при решении практических задач в той области, в которой они обучаются. Система может быть применена не только в обучении, но и в практической деятельности.

Система MATLAB выполняет операции над векторами и матрицами. Одномерный массив называют *вектором*, а двумерный – *матрицей*: векторы из 4-х элементов имеют структуру [1 2 3 4] или [1, 2, 3, 4];

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 8 & 7 & 6 \end{pmatrix}$$

это матрица размером 3x4, т. е. содержащая 3 строки и 4 столбца с числовыми данными;

$$\begin{pmatrix} a & a + b & a + b/c \\ x & yx & z \\ 1 & 2 & 3 \end{pmatrix}$$

а это матрица размером 3x3 с элементами разного типа.

Массивы в общем случае характеризуются размерностью и размером.

*Размерность* массива определяет его структурную организацию в виде одной строки или одного столбца (размерность 1), страницы (размерность 2), куба (размерность 3) и т. д. MATLAB допускает задание и использование многомерных массивов ряда типов, в том числе массивов ячеек и записей.

*Размер* вектора — это число его элементов, а размер матрицы определяется числом ее строк  $m$  и столбцов  $n$ . Обычно размер матрицы указывают как  $m \times n$ . Матрица называется квадратной, если  $m = n$ , то есть число строк матрицы равно числу ее столбцов. Многие элементы *разреженных матриц* — нули. Поэтому для эффективной работы с такими матрицами имеется ряд специальных функций.

Векторы и матрицы могут иметь имена, например,  $V$  — вектор или  $M$  — матрица. Имена векторов и матриц набираются **полужирным** шрифтом. Элементы векторов и матриц рассматриваются как *индексированные переменные*. Например,  $V_2$  — второй элемент вектора  $V$ ;  $M_{2,3}$  — третий элемент матрицы  $M$ , расположенный во второй строке.

Интересно отметить, что даже обычные числа и переменные в MATLAB рассматриваются как матрицы размером  $1 \times 1$ , что даст единообразные формы и методы проведения операций над обычными числами и массивами. Это также означает, что большинство вычислительных функций могут работать с аргументами в виде векторов и матриц, вычисляя значения для каждого их элемента.

### Файловая система MATLAB

Система MATLAB состоит из многих тысяч файлов, находящихся в множестве папок. *Файл* — это некоторая совокупность данных в широком понимании, хранящихся в памяти компьютера (обычно на дисковых накопителях) и объединенных под некоторым *именем файла*, после которого через точку указывается *расширение файла*. Файлы могут содержать машинные коды (исполняемые файлы), тексты (текстовые файлы), исходные данные для математических расчетов и результаты их выполнения.

В MATLAB особое значение имеют файлы двух типов — с расширениями **.mat** и **.m**. Первые являются бинарными файлами, представляющими запись сеанса (сессии) работы системы. Вторые представляют собой текстовые файлы, содержащие внешние определения команд и функций системы. Именно к ним относится большая часть команд и функций, в том числе задаваемых пользователем для решения своих специфических задач. Нередко встречаются и файлы с расширением **.c** (на языке C), файлы с откомпилированными кодами с расширением **.mex** и др. Исполняемые файлы имеют расширение **.exe**.

Часть примеров, которые могут выполняться, отмечены слева жирной линией, как показано в этом абзаце.

## 2.2. Интерфейс MATLAB

Система постоянно развивается, меняется интерфейс пользователя. Однако основные окна остаются без изменения, меняется только их размещение на экране. Здесь мы будем рассматривать версию MATLAB 2009.

При запуске системы появляется окно, изображенное на рис. 2.1.

В центральной части находится командное окно (Command Window), а в нем — командная строка, в которую вводятся команды и выражения. Для ввода и корректировки выражений в ней содержится простейший строчный редактор. Не перечисляя всех его команд, отметим только две — стрелка вверх и стрелка вниз, с помощью которых можно перелистывать команды, введенные ранее. В остальном редактирование похоже на редактирование строки в редакторе текстов Word.

Существуют также команды управления окном, среди которых отметим **clc** — очистка экрана; **more on** и **more off** — включение и выключение постраничного вывода, полезны при выводе больших текстов m-файлов.

Вычисления можно производить в режиме прямых вычислений и в режиме выполнения команд путем запуска m-файлов.

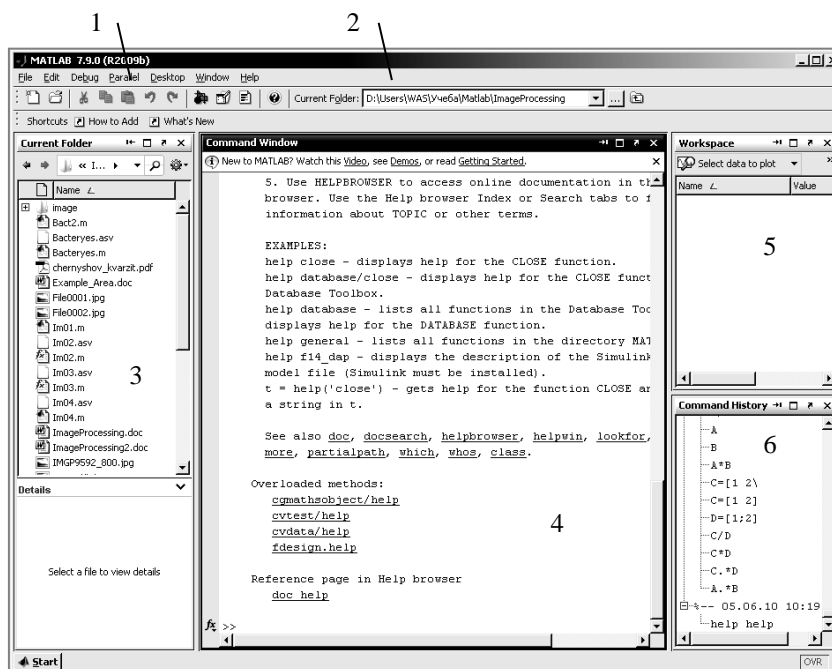


Рис. 2.1. Оконный интерфейс системы MATLAB

1 – основное меню; 2 - установка текущего каталога; 3 – содержимое текущего каталога; 4 – командное окно; 5 – содержимое рабочего пространства; 6 – история команд

В режиме прямых вычислений команды вместе с выражениями записываются в командную строку, которая отмечается указателем `>>`. После записи выражений и команд для выполнения следует нажать клавишу Enter. При работе с командной строкой используется ряд правил:

- для указания ввода исходных данных используется символ `>>`; если этого символа нет, то значит выполняются вычисления, и прежде чем вводить новые команды и выражения, следует подождать до завершения предыдущей операции и появления этого символа;
- данные вводятся с помощью простейшего строчного редактора;
- для блокировки вывода результата вычислений некоторого выражения после него надо установить знак `;` (точка с запятой);
- если не указана переменная со значением результата вычислений, то MATLAB назначает такую переменную с именем **ans**;
- знаком присваивания является привычный математикам знак равенства `=`, а не комбинированный знак `:=`, как во многих других математических системах;
- встроенные функции (например, **sin**) записываются строчными буквами, и их аргументы указываются в *круглых скобках*;
- результат вычислений выводится в строках вывода (без знака `>>`);
- диалог происходит в стиле «задал вопрос, получил ответ».

Пример некоторых простых вычислений:

```
>> 2+3
```

```

ans
    5

>> x=sin(1)
x =
    0.8415

>> V=[1 2 3 4]
V =
    1 2 3 4

>> sin(V)
ans=
    0.8415  0.9093  0.1411 -0.7568

>> exp(V)
ans=
    2.7183  7.3891  20.0855 54.5982

```

Обратите внимание на форму ответов при выполнении простых операций без указания переменной, которой присваивается результат.

В некоторых случаях вводимое математическое выражение может оказаться настолько длинным, что для него не хватит одной строки. В этом случае часть выражения можно перенести на новую строку с помощью знака многоточие ... (3 или более точек).

При осуществлении повторяющихся вычислений текст программы целесообразнее набирать с помощью редактора, который вызывается из командной строки с помощью команды **Edit**. В этом случае осуществляется вычисление по программе из редактора. Такая программа может быть сохранена в виде m-файла, а если при этом в первой строке указать, что это функция, то она затем может использоваться в виде подпрограммы-функции с соответствующей передачей данных в нее и обратным получением (возвращением) результатов выполнения.

Программа может быть откомпилирована и тем самым преобразована в исполняемый (загрузочный) модуль, который позволяет выполнять ее без загрузки системы MATLAB, что является третьей формой осуществления вычислений.

### 2.3. Типы данных системы MATLAB

#### Числа целые и вещественные

Большое значение имеют типы данных, с которыми работает система MATLAB. Числа в MATLAB представляются в обычном виде целых, дробных и вещественных чисел, например

```
2 -3 2.301 0.00001 123.456e-24 -234.456e10
```

Как нетрудно заметить, в мантиссе чисел целая часть отделяется от дробной не запятой, а точкой, что принято в большинстве языков программирования. В системе MATLAB этот принцип сохраняется, даже если в операционной системе Windows в разделе «Язык и стандарты» в качестве десятичного разделителя установлена запятая. Для отделения порядка числа от мантиссы используется символ *e*. Знак «плюс» у чисел не проставляется, а знак «минус» у числа называют *унарным минусом*. Пробелы между символами в числах не допускаются.

### Форматы чисел

По умолчанию MATLAB выдает числовые результаты в нормализованной форме с четырьмя цифрами после десятичной точки и одной до нее. Операции над числами выполняются в формате, который принято называть форматом чисел с двойной точностью. Для установки формата представления чисел используется команда

>> FORMAT NAME

где **NAME** — имя формата. Для иллюстрации различных форматов рассмотрим вектор, содержащий два элемента-числа:

$x = [4/3 \quad 1.2345e-6]$

В различных форматах их представления будут иметь следующий вид:

Формат	Пример 1	Пример 2
format short	1.3333	0.0000
format short e	1.3333E+000	1.2345E-006
format long	1.3333333333333338	0.000001234500000
format long e	1.3333333333333338E+000	1.2345000000000000E-006
format bank	1.33	0.00

Задание формата сказывается только на форме вывода чисел. Вычисления все равно происходят в формате двойной точности, а ввод чисел возможен в любом удобном для пользователя виде.

Числа могут быть комплексными:  $z = \text{Re}(z) + \text{Im}(z) \cdot i$ . Такие числа содержат действительную  $\text{Re}(z)$  и мнимую  $\text{Im}(z)$  части. Мнимая часть имеет множитель  $i$  или  $j$ , означающий корень квадратный из  $-1$ , например:

$3i$   
 $2j$   
 $-3.141i$   
 $-123.456+2.7e-3$

$2+3i$

Функция **real(z)** возвращает (имеет своим результатом) действительную часть комплексного числа  $\text{Re}(z)$ , а функция **imag(z)** — мнимую,  $\text{Im}(z)$ .

### Константы и системные переменные

*Константа* — это предварительно определенное числовое или символьное значение, представленное уникальным именем. Числа (например, 1, -2 и 1,23) являются безымянными *числовыми константами*.

Другие виды констант 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:

- $i$  или  $j$  — мнимая единица (корень квадратный из -1);
- $\pi$  - число  $\pi = 3.1415926\dots$ ;
- $\text{eps}$  — погрешность для операций над числами с плавающей точкой ( $2^{-52}$ );
- $\text{real min}$  — наименьшее число с плавающей точкой ( $2^{-1022}$ );
- $\text{real max}$  — наибольшее число с плавающей точкой ( $2^{1021}$ );
- $\text{inf}$  — значение машинной бесконечности;
- $\text{ans}$  — переменная, хранящая результат последней операции и обычно вызывающая его отображение на экране дисплея;
- $\text{NaN}$  — указание на нечисловой характер данных (Not-a-Number).

Вот примеры применения системных переменных:

```
|| >> 2*pi  
ans =  
    6.2832
```

```
|| >> eps  
ans =  
    2.2204e-016
```

```
|| >> real min  
ans =  
    2.2251e-308
```

```
|| >> real max  
ans =  
    1.7977e+308
```

```
|| >> 1/0
Warning: Divide by zero.
ans =
     Inf
```

```
|| >> 0/0
Warning: Divide by zero.
ans =
     NaN
```

Здесь **Warning** означает предупреждение.

### Строки и текстовые комментарии

*Символьная константа* — это цепочка символов, заключенных в апострофы, например:

```
'Hello, world'
'2+3'
```

Последнее выражение вычисляться не будет, т. к. представляет собой символьную константу. Комментарии в системе обозначаются знаком процента:

```
% Это комментарий.
```

Операторы и функции в системе MATLAB сходны с другими языками, поэтому здесь мы не будем на них останавливаться. Особо отметим применение оператора `:` (двоеточие).

### Применение оператора `:` (двоеточие)

Очень часто необходимо произвести формирование упорядоченных числовых последовательностей. Такие последовательности нужны для создания векторов или значений абсциссы при построении графиков. Для этого в MATLAB используется оператор `:` (двоеточие):

```
Начальное_значение:Шаг:Конечное_значение
```

Данная конструкция порождает последовательность чисел, которая начинается с начального значения, идет с заданным шагом и завершается конечным значением. При этом действуют следующие правила:

```
Начальное_значение < Конечное_значение Шаг > 0
Начальное_значение > Конечное_значение Шаг < 0
```



Если Шаг не задан, то он принимает значение 1. Примеры применения оператора : даны ниже:

```
|| >> 1:5
ans =
    1 2 3 4 5
```

```
|| >> l=0:2:10
l =
    0 2 4 6 8 10
```

```
|| >> j=10:-2:2
j =
   10 8 6 4 2
```

```
|| >> V=0:pi/2:2*pi:
>> V
V =
    0 1.5708 3.1416 4.7124 6.2832
```

```
|| >> X=1:-.2:0
X =
    1.0000 0.8000 0.6000 0.4000 0.2000 0
```

Переменные с множеством упорядоченных значений принято называть *ранжированными*.

Как уже отмечалось, принадлежность MATLAB к матричным системам вносит коррективы в назначение операторов и приводит, при неумелом их использовании, к казусам. Рассмотрим следующий характерный пример:

```
|| >> x=0:5
x =
    0 1 2 3 4 5
```

```
|| >> cos(x)
ans =
    1.0000 0.5403 -0.4161 -0.9900 -0.6536 0.2837
```

```
|| >> sin(x)/x
ans =
    0.0862
```

Вычисление массива косинусов здесь прошло корректно. А вот вычисление массива функции  $\sin(x)/x$  даст на первый взгляд неожиданный эффект — вместо массива с шестью элементами вычислено единственное значение.

Причина «парадокса» здесь в том, что оператор `/` вычисляет отношение двух матриц, векторов или массивов. Если они одной размерности, то результат будет одним числом, что в данном случае и выдала система. Чтобы действительно получить массив значений  $\sin(x)/x$ , надо использовать специальный оператор почленного деления массивов `./`. Тогда будет получен массив чисел:

```
|| >> sin(x)./x
Warning: Divide by zero.
ans =
    NaN 0.8415 0.4546 0.0470 -0.1892 -0.1918
```

Впрочем, и тут без особенностей не обошлось. Так, при  $x = 0$  значение  $\sin(x)/x$  дает устранимую неопределенность вида  $0/0 = 1$ . Однако, как и всякая численная система, MATLAB классифицирует попытку деления на 0 как ошибку и выводит соответствующее предупреждение. А вместо ожидаемого численного значения выводится символьная константа NaN.

Выражения с оператором `:` могут использоваться в качестве аргументов функций для получения множественных их значений. Проиллюстрируем это вычислениями функции Бесселя. Формат соответствующей функции MATLAB

```
bessel(nu,z)
```

где **nu** – порядок, **z** – значение аргумента.

В приводимом ниже примере вычислены функции Бесселя порядка от 0 до 5 со значением аргумента  $x = 1/2$ :

```
|| >>bessel(0:1:5,1/2)
ans =
    0.9385 0.2423 0.0306 0.0026 0.0002 0.0000
```

А в следующем примере вычислено шесть значений функции Бесселя нулевого порядка для значений аргумента от 0 до 5 с шагом 1:

```
|| >>bessel(0,0:1:5)
ans =
    1.0000 0.7652 0.2239 -0.2601 -0.3971 -0.1776
```

## 2.4. Простейшие приемы работы с векторами и матрицами

### Особенности задания векторов и матриц

Для задания вектора надо перечислить значения его элементов в квадратных скобках, разделяя их пробелами или запятыми (см. примеры выше). Задание матрицы требует указания различных строк. Для различения строк используется знак ; (точка с запятой). Этот же знак в конце ввода предотвращает вывод матрицы или вектора на экран дисплея. Так, ввод

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

задает квадратную матрицу, которую можно вывести

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

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

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

### Доступ к отдельным элементам

Для указания отдельного элемента вектора или матрицы их рассматривают как *индексированные переменные*. Для этого используются выражения вида V(i) или M(1,j). При этом индексы разделяются запятыми. Например, если задать

```
>> M=[1 2; 4 5];
>>M(2,2)
ans =
     5
```

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

$$M(i;j)=x$$

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

```
>>M(2,2)=10
```

Выражение  $M(i)$  с одним индексом дает доступ к элементам матрицы, развернутым в один столбец. Такая матрица образуется из исходной, если подряд сверху вниз выписать ее столбцы. Следующие примеры поясняют такой доступ к элементам матрицы  $M$

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

```
|| >>M(2)
ans =
    4
```

```
|| >>M(8) ans =
    6
```

```
|| >>M(9)
ans =
    9
```

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

Возможно задание векторов и матриц с комплексными элементами, например

```
>> i-sqrt(-1);
```

```
>> CM = [1 2; 3 4] + i*[5 6; 7 8]
```

или

```
>> CM = [1+3*i 2+6*i; 3+7*i 4+8*i]
```

Это создает матрицу

$$\begin{bmatrix} 1+3i & 2+6i \\ 3+7i & 4+8i \end{bmatrix}$$

Наряду с операциями над отдельными элементами матриц и векторов система позволяет производить операции умножения, деления и возведения в степень сразу над всеми элементами, то есть над массивами. Для этого перед операцией ставится точка. Например, оператор \* означает знак умножения для векторов или матриц, а оператор .\* — почленное умножение всех элементов массива. Так, если M — матрица, то M.\*2 даст матрицу, все элементы которой умножены на скаляр — число 2. Впрочем, для умножения матрицы на скаляр оба выражения — M\*2 и M.\*2 — оказываются равноценными.

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

```

>> M=magic(4)
M =
    16  2  3 13
     5 11 10  8
     9  7  6 12
     4 14 15  1

```

```

>> sum(M)
ans =
    34 34 34 34

```

```

>> sum(M')
ans =
    34 34 34 34

```

```

>> sum(diag(M))
ans =
    34

```

```

>> M(1,2)+M(2,2)+M(3,2)+M(4,2)
ans =
    34

```

### Удаление столбцов и строк матриц

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

этого используются пустые квадратные скобки [ ]. Прделаем это над матрицей M.

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

```
M =
```

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

Удалим второй столбец:

```
||>>M(:,2)=[ ]
```

```
M =
```

```
1 3
4 6
7 9
```

А теперь удалим вторую строку:

```
||>> M(2,:)=[]
```

```
M =
```

```
1 3
7 9
```

### Сохранение значений векторов и матриц при завершение работы с системой

Значения матриц и векторов могут быть сохранены при завершении работы. Для завершения работы с системой можно использовать команды **quit**, **exit** или комбинацию клавиш **Ctrl+Q**. Если необходимо сохранить значения всех переменных (векторов, матриц) системы, то перед этим следует дать команду **save** нужной формы. Команда **load** после загрузки системы считывает значения этих переменных и позволяет начать работу с системой с того момента, когда она была прервана.

## 2.5. Операторы и функции

*Оператор* — это специальное обозначение для определенной операции над данными — *операндами*. Например, простейшими арифметическими операторами являются знаки суммы +, вычитания -, умножения \* и деления /. Операторы используются совместно с операндами. Например, в выражении 2+3 знак + является оператором сложения, а числа 2 и 3 — операндами.

Большинство операторов относится к матричным операциям, что может служить причиной серьезных недоразумений. Например, операторы умножения \* и деления / вычисляют произведение и частное от деления двух многомерных массивов, векторов или матриц. Есть ряд специальных операторов,

например, оператор `\` означает деление *справа налево*, а операторы с точкой `.*` и `./` означают соответственно *поэлементное* умножение и *поэлементное* деление массивов.

*Функции* — это имеющие уникальные имена объекты, выполняющие определенные преобразования своих аргументов и при этом возвращающие результаты этих преобразований. *Возврат результата* — отличительная черта функций. При этом результат вычисления функции с одним выходным параметром подставляется на место ее вызова, что позволяет использовать функции в математических выражениях, например, функцию **sin** в `2*sin(pi/2)`.

Функции в общем случае имеют список аргументов (параметров), заключенный в круглые скобки. Например, функция Бесселя записывается как **bessel(NU,X)**. В данном случае список параметров содержит два аргумента — NU в виде скаляра и X в виде вектора. Многие функции допускают ряд форм записи, отличающихся списком параметров. Если функция возвращает несколько значений, то она записывается в виде `[Y1, Y2....]=func(X1, X2...)`, где Y1, Y2,... — список *выходных* параметров и X1, X2.... — список *входных* аргументов (параметров).

Со списком элементарных функций можно ознакомиться, выполнив команду **help elfun**, а со списком специальных функций — с помощью команды **help specfun**. Функции могут быть *встроенными* (внутренними) и *внешними*, или *m-функциями*. Так, встроенными являются наиболее распространенные элементарные функции, например, **sin(x)** и **exp(y)**, тогда как функция **sinh(x)** является внешней функцией. Внешние функции содержат свои определения в m-файлах. Встроенные функции хранятся в откомпилированном ядре системы MATLAB, в силу чего они выполняются предельно быстро.

## 2.6. Работа со справочной системой MATLAB

При изучении математических систем, особенно самостоятельно, трудно переоценить роль справочной системы. В MATLAB используется развитая (многоуровневая) справочная система, использующая все известные приемы получения справочных сведений - от прямого запроса в командной строке до использования гипертекстовых справочных подсистем и документов в формате PDF.

### Вызов списка примеров интерактивной справки

MATLAB имеет интерактивную справочную систему, которая реализуется в командном режиме с помощью ряда команд. Одной из них является команда

```
>> help
```

которая выводит весь список папок (каталогов), содержащих m-файлы с определениями операторов, функций и иных объектов. Этот внушительный список

дает наглядное представление о пакетах прикладных программ, расширяющих возможности системы MATLAB и содержащих массу серьезных примеров применения системы.

### Справка по конкретному объекту

Для получения справки по какому-либо конкретному объекту используется команда

```
>> help имя
```

где имя — имя объекта, для которого требуется вывод справочной информации. Мы уже приводили пример помощи по разделу операторов **ops**. Ниже дается пример для функции вычисления гиперболического синуса, намеренно введенного с неверным указанием имени:

```
>> help hsln  
hsln.m not found.
```

Нетрудно заметить, что система помощи сообщает, что для функции с именем **hsln** соответствующий m-файл отсутствует. Если ввести имя верно, получится вот что:

```
>> help sinh  
SINH Hyperbolic sine.  
SINH(X) is the hyperbolic sine of the elements of X. Overloaded meth-  
ods  
help sym/sinh.m
```

Теперь полученное сообщение будет содержать информацию о функции гиперболического синуса **sinh**. Хотя имя функции в MATLAB задается маленькими (строчными) буквами, в сообщениях справочной системы имена функций и команд выделяются большими (прописными) буквами.

### Справка по определенной группе объектов

Пользователя системы MATLAB часто интересует набор функций, команд или иных понятий, относящихся к определенной группе объектов. Ниже дан пример вызова справки по группе объектов **timefun**.

```
>> help timefun  
Time and dates.  
Current date and time.  
now - Current date and time as date number.  
date - Current date as date string.
```



clock	- Current date and time as date vector.
Basic functions.	
datenum	- Serial date number.
datestr	- String representation of date.
datevec	- Date components.
Date functions.	
calendar	- Calendar.
weekday	- Day of week.
eomday	- End of month.
datetick	- Date formatted tick labels.
Timing functions.	
cputime	- CPU time in seconds.
tic, toe	- Stopwatch timer.
etime	- Elapsed time.
pause	- Wait in seconds.

После уточнения состава определенной группы объектов можно получить детальную справку по любому выбранному объекту. Как это делается, было описано выше.

Если возникают затруднения с выбором объекта или группы объектов, то можно вывести содержимое системы **help**, введя одно слово **help**. После просмотра содержимого, можно подобрать необходимые объекты или группы объектов, содержащиеся в справочной системе. Например, при попытке получения справки по всем нейронным сетям ввод **help neural network** дает справку только по разделу **network**. Поэтому следует ввести **help**, после вывода содержания можно найти, что здесь нейронные сети обозначены как **nnet**. После этого ввод **help nnet** дает ожидаемый результат – вывод справки по операторам и функциям для работы с нейросетями.

### Справка по ключевому слову

Ввиду обилия m-функций в системе MATLAB, число которых около 800, важное значение имеет поиск m-функций по ключевым словам. Для этого служит команда **lookfor** **Ключевое слово** или **lookfor** **Ключевые слова**.

В первом случае ищутся все m-файлы, в заголовках которых встречается заданное ключевое слово, и заголовки обнаруженных файлов выводятся на экран. Следует отметить, что широкий поиск по одному ключевому слову может подчас привести к выводу многих десятков определений и длиться довольно долго.

Для уточнения и сокращения поиска следует использовать вторую форму команды **lookfor**. Вот пример ее применения (поиск осуществляется долго):

```
>> lookfor 'neural network'
NEURDEMO Financial EXPO Neural Network demonstration.
nnetlin Linear Neural Network demo for Financial Expo
```

NEWGRNN Design a generalized regression neural network.

и т. д. – довольно большой список.

Число найденных определений зависит от того, с каким набором пакетов прикладных программ (расширений) поставляется версия системы MATLAB.

## 2.7. Работа с файлами данных

При обработке данных они хранятся в файлах, поэтому операции чтения данных из файла и записи в файл являются одними из основных.

Получить справку по этому разделу можно, набрав **help MATLAB\iofun**.

Для нас наиболее важными будут функции ввода/вывода, работающие с текстовыми файлами. Это **dlmread** при вводе и **dlmwrite** при записи на диск, здесь **dlm** означает **delimited**, т. е. разделенные каким-либо разделителем (пробелом, запятой и т. д.).

Пример использования оператора **dlmread**:

```
dlmread('myfile.txt')
```

Эта функция читает данные из текстового файла с именем **myfile.txt**. Заметим, что текстовый файл может быть подготовлен с помощью обычного Блокнота в системе Windows. Кроме того, такие файлы часто образуются при регистрации в процессе какого-либо эксперимента, либо могут быть преобразованы из бинарных форматов в текстовые с помощью специальных конверторов.

Данные могут быть загружены из \*.mat-файла, в котором они хранятся в двоичном (бинарном) формате. Это делается с помощью оператора **load**, например

```
D=load('textpt.mat')
```

В системе MATLAB существуют возможности для импорта файлов из других форматов. Например, для импорта из формата **Excel** используется функция **xlsread**

```
C=xlsread('myfile.xls')
```

## КОНТРОЛЬНЫЕ ВОПРОСЫ К ГЛАВЕ 2

1. Что означает название системы MATLAB?
2. Перечислите названия основных составляющих оконного интерфейса системы MATLAB, поясните выполняемые ими функции.
3. Дайте определения и приведите примеры векторов и матриц, используемых в системе MATLAB.
4. Поясните различие понятий «размерность» и «размер» матриц.
5. Какие типы файлов используются в системе MATLAB, в чем особенность каждого типа?
6. Что такое «прямые вычисления», как они осуществляются, в чем их отличие от вычислений по программе, как осуществляются последние?
7. Как создаются программные функции?
8. Можно ли выполнять программы, созданные в MATLAB, без запуска самой системы?
9. С какими типами данных работает система MATLAB?
9. Какие форматы чисел используются в системе MATLAB? Приведите примеры.
10. В чем различие символьных констант и текстовых комментариев?
11. Каково назначение оператора : (двоеточие)? Приведите примеры применения.
12. Что такое ранжированная переменная?
13. В каких случаях деление друг на друга двух векторов или матриц даст одно значение, а в каких – матрицу значений, содержащую несколько элементов?
14. Каким образом возможно получать доступ к элементам, строкам, столбцам матрицы?
15. В чем разница операторов \* и .\*; / и ./, что означает оператор \?
16. Как удалять строки или столбцы матрицы?
17. Что такое операнды, команды, выражения, операторы, функции? Приведите примеры.
18. Как вызывать справку? Перечислите, в какой последовательности следует вызывать справку, если требуется произвести какое-либо действие, а функция или оператор, с помощью которых это действие осуществляется, неизвестны.
19. Перечислите наиболее употребительные функции ввода-вывода, используемые для работы с файлами.
20. Как осуществить импорт данных из Excel-файла в MATLAB?

## 3. ГРАФИКА И ГРАФИЧЕСКИЕ ФОРМАТЫ

### 3.1. Основные понятия

**Цифровое изображение (ЦИ)** - модель реального или синтезированного изображения, хранящаяся на машинном носителе в виде совокупности цифровых кодов.

Любое изображение на компьютере может быть представлено в двух графических режимах:

- в векторном виде;
- в растровом виде.

На практике используются как растровые, так и векторные модели. Векторные модели применяются в процессе создания синтезированных изображений, а растровые – при работе с естественными изображениями, получаемыми в результате сканирования, фото-, видеосъемки объектов окружающей нас действительности. Векторные изображения сравнительно легко могут быть конвертированы в растровые, а конвертация растровых изображений в векторные затруднительна.

Рассмотрим понятия, относящиеся в первую очередь к растровым изображениям.

**Пиксель** (англ. pixel - picture element - элемент картинки) - неделимый прямоугольный элемент растровой модели, параметры которого описывают соответствующий ему участок реального или синтезированного изображения.

**Параметры растровых изображений:**

- размер - произведение ширины на высоту в пикселях;
- разрешающая способность (разрешение) - количество информации на единицу длины. Измеряется в ppi (pixels per inch — пиксели на дюйм) и dpi (dots per inch - точки на дюйм). Имеет смысл только, если известны реальные размеры изображения или отпечатка. Так, например, для представления изображения на экране монитора при сканировании в масштабе 1:1 следует задавать разрешение 72 ppi - типовое значение разрешающей способности большинства мониторов PC и 95 ppi - для MAC.

**Модель** - способ описания элементов изображения в цифровом виде. Например, Bitmap, Grayscale, Indexed, RGB, HLS, Lab, CMYK.

**Цветовая модель RGB** - естественный язык цвета сканеров, мониторов и других электронных устройств, название образовано от слов Red (красный), Green (зеленый), Blue (синий).

**Глубина цвета** - количество бит памяти, выделяемых для описания тоновых или цветовых характеристик каждого пикселя в соответствие с моде-

лю. Например, 1бит/пикс. (Bitmap, Halftone), 8 бит/пикс. (Grayscale, Indexed), 24 бит/пикс. (RGB).

**Формат файла** - способ организации информации в файле. Графические файлы служат для хранения изображений между сеансами работы с графическими программами и переноса изображений между программами и компьютерами. Графическая информация в файлах кодируется несколько иначе, чем в памяти компьютера. Более того, способов кодирования, называемых форматами, существует множество. Существование большого числа форматов графических файлов обусловлено специфическими сферами их применения. На практике часто используются следующие основные форматы со сжатием информации:

- BMP, TIFF – универсальные форматы без сжатия;
- GIF - для хранения рисунков и анимаций (недеструктивное сжатие, т. е. сжатие без разрушения изображения);
- JPEG - для хранения фотографий (деструктивное сжатие);
- PNG8 - для рисунков и фотографий в моделях Grayscale и Indexed (недеструктивное сжатие);
- PNG24 - для рисунков и фотографий в модели RGB (недеструктивное сжатие).

Файлы, в которые записаны изображения одного и того же размера, но в разных форматах, будут иметь различную величину. Наименьший размер будут иметь форматы с деструктивным сжатием изображений, но качество их будет хуже, чем без сжатия.

### 3.2. Параметры графических форматов

**Распространенность.** Многие приложения имеют собственные форматы файлов. Они поддерживают особые возможности конкретных программ, но могут оказаться несовместимыми с другими приложениями. Программы иллюстрирования и издательские системы могут не уметь импортировать такие форматы или делать это некорректно. Вопрос распространенности касается не только собственных форматов программ. Некоторые форматы разрабатывались специально под аппаратное обеспечение (например, форматы Scitex, Targa, Amiga IFF). Подобные форматы лучше не использовать. Сохранение изображений в малораспространенных форматах может создать потенциальные проблемы при переносе их на другие компьютеры.

**Соответствие сфере применения.** Большинство графических форматов ориентировано на конкретные области применения. В случае ошибки при выборе формата изображение может оказаться непригодным для использования. Например, сохранение изображения в формате JPEG с большим коэффициентом

том сжатия делает его непригодным для печати из-за потери качества. При этом повторное открытие и сохранение в другом формате не исправит допущенную ошибку.

**Поддерживаемые типы точечных изображений и цветовые модели.** Следует выбирать формат файлов, поддерживающий заданные сферой применения типы изображений. Например, формат ВМР не поддерживает изображений в модели СМУК, требующейся в полиграфии, и, следовательно, не может использоваться в этой сфере. Тем не менее, следует учитывать возможность последующего преобразования типов и цветовых моделей, требуемых в выбранной сфере применения.

**Возможность сжатия графической информации.** Для уменьшения размеров графических файлов многие форматы предполагают сжатие данных. Выбор одного из таких форматов экономит место на жестком диске и тех носителях, которые используются для хранения и передачи файлов.

**Способ сжатия.** Форматы файлов, поддерживающие сжатие, используют для этого различные алгоритмы. Все алгоритмы сжатия делятся на те, что не приводят к потерям качества, и те, что снижают качество изображений. Последние позволяют достичь на порядок более высоких коэффициентов сжатия.

Часто возникает необходимость минимизации объема файла, а значит времени его передачи по сети при допустимом уровне качества. Достигается использованием соответствующего формата файла (GIF, JPEG или PNG) и подбором совокупности параметров, определяющих наилучшее соотношение качество/размер.

### **3.3. Два режима представления графической информации**

Рассмотрим более подробно режимы представления изображений. Как было сказано выше, любое изображение на компьютере может быть представлено в двух графических режимах:

- в векторном виде;
- в растровом виде.

#### **Векторная графика (Vector drawing)**

Форматы: .CDR, .AI и др.

Это вид кодировки графических изображений, основанный на геометрии, но не точек (как в растровой графике), а кривых, описываемых сплайнами. В качестве сплайнов выбраны кривые Безье<sup>1</sup>. Обычно используются ку-

---

<sup>1</sup> Пьер Безье - французский математик, рассчитывал сплайны корпусов автомобилей.

бичные сплайны, описываемые полиномами третьей степени, которые задаются 4 коэффициентами.

**Сплайн** - основное понятие векторной графики, на русский язык переводится как «линейка». Это особая кривая, плавно проходящая через заданные точки. Изображения линий в векторной графике строятся с помощью сплайнов. На сплайнах построены современные шрифты **TrueType** и **PostScript**. Сплайн, которым описывается элементарная кривая, можно построить, зная четыре коэффициента  $P_0$ ,  $P_1$ ,  $P_2$  и  $P_3$ , соответствующие четырем точкам на плоскости. Перемещая эти точки, можно менять форму кривой.

Примером векторной графики служат работы, созданные в графическом редакторе CorelDraw. В отличие от раstra векторное изображение состоит из отдельных линий-направляющих (векторов) которые образуют изображение. В файле хранится информация не о каждой точке, а об элементах, из которых состоит изображение, т. е. о направляющих, из которых она создана. Векторные изображения занимают сравнительно небольшой объем и легки в редактировании. Любой элемент картинки может быть изменен отдельно от других. Изображение легко меняет размер, не теряя качества и сохраняя первоначальную композицию (расположение элементов). Вектор пластичен, что позволяет отображать его на устройствах с различной разрешающей способностью одинаково качественно. Но изображения векторной графики просты по визуальному восприятию и в основном выглядят «нарисованными».

### **Достоинства векторной графики**

1. Малый объем памяти. При кодировании векторного изображения хранится не само изображение объекта, а координаты четырех точек, поэтому объем памяти очень мал по сравнению с точечной графикой. Векторная графика - очень экономичный способ кодирования. Она экономна в плане объемов дискового пространства, необходимого для хранения изображений: это связано с тем, что сохраняется не само изображение, а только некоторые основные данные, используя которые программа всякий раз воссоздает изображение заново. Кроме того, описание цветовых характеристик несильно увеличивает размер файла.
2. Свобода трансформации. Векторное изображение можно вращать, масштабировать без потери качества изображения. Объекты векторной графики просто трансформируются, и ими легко манипулировать, что не оказывает практически никакого влияния на качество изображения.
3. Аппаратная независимость. Векторная графика «работает» с идеальными объектами, которые сами приноравливаются к изменениям: можно не знать, для каких устройств делается тот или иной документ. Векторная графика максимально использует возможности разрешающей способно-

сти любого выводного устройства: изображение всегда будет настолько качественным, насколько способно данное устройство.

### **Недостатки векторной графики**

1. Программная зависимость. Каждая программа строит кривые Безье по своим алгоритмам. Например, формат \*.CDR программы CorelDraw не описан и является нестандартным. Часто необходимо конвертирование. Каждая программа сохраняет данные в своем собственном формате, поэтому изображение, созданное в одном векторном редакторе, как правило, не конвертируется в формат другой программы без погрешностей.
2. Сложность векторного принципа описания изображения не позволяет автоматизировать ввод графической информации и сконструировать устройство, подобное сканеру, для растровой графики.
3. Векторная графика действительно ограничена в чисто живописных средствах и не предназначена для создания фотореалистических изображений.

### **Растровая графика (Raster drawing)**

Изображения в растровой графике состоят из отдельных точек (пикселей) различных цветов, образующих цельную картину (наподобие мозаики). Типичным примером растровой графики служат отсканированные фотографии или изображения - один из примеров представлен на рис. 3.1.

Применение растровой графики позволяет добиться изображения высочайшего фотореалистичного качества. Но такие файлы очень объемны и трудно редактируемы. При изменении размеров качество изображения ухудшается. Так, при уменьшении исчезают мелкие детали, а при увеличении картинка превращается в набор пикселей. При печати растрового изображения или при просмотре его на устройствах, имеющих недостаточную разрешающую способность, значительно ухудшается восприятие образа.

В зависимости от того, сколькими двоичными символами кодируется цвет каждого пикселя, изображение (объект) может быть бинарным (т. е. штриховым, черно-белым), серым и цветным.



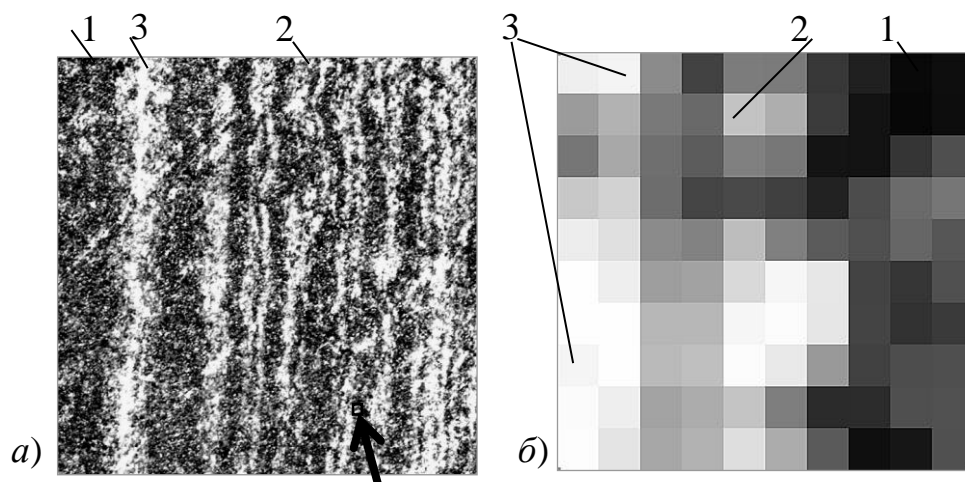


Рис. 3.1. Растровое изображение размером  $628 \times 628$  пикселей скана аншлифа железистого кварцита (а), а также вырезанный (отмечен стрелкой внизу справа) и увеличенный участок этого изображения  $10 \times 10$  пикселей, на котором отчетливо видны отдельные элементы (б), здесь различаются области магнетита (1), гематита (2), кварца (3)

При *бинарном кодировании* в каждую ячейку изображения записывают либо 0, либо 1. Нули соответствуют черным точкам, а единицы – белым. Если надо закодировать какое-то изображение, содержащее интересующий объект белого цвета на черном фоне, то на него «накладывают» сетку и создают матрицу (таблицу) того же размера, что и сетка, заполняя единицами ячейки, наложенные на объект, и нулями вне объекта. В ячейки сетки, заполненные частично, можно записать 1, если она заполнена наполовину или более, и 0, если она заполнена менее чем наполовину. Если эту матрицу вывести на экран или принтер или на диск для хранения, то получим оттиск объекта, при этом объект будет белого цвета, а фон – черного.

*Серое изображение* в каждой ячейке содержит уже не 0 или 1, а числа, характеризующие яркость этого пикселя, кодируется такое изображение одним или двумя байтами.

*Цветное изображение* для описания каждого пикселя использует три ячейки, в каждой из которых записаны яркости соответствующих основных цветов, из которых формируется цвет пикселя.

Детальность представления изображения характеризуется разрешающей способностью, показывающей либо максимальное количество элементов изображения на единицу длины, либо минимальное расстояние между такими элементами.

Разрешающая способность изображения измеряется в единицах:

- **ppi (pixel per inch** - пиксель на инч (дюйм)) - количество пикселей на единицу длины в 1 дюйм;
- **dpi (dots per inch** - точки на дюйм) - количество точек на единицу длины в 1 дюйм.

При этом следует учитывать, что 1 дюйм = 25,4 мм.

### **Достоинства растровой графики**

1. Каждый пиксель независим друг от друга.
2. Техническая реализуемость автоматизации ввода (оцифровки) изображительной информации. Существует развитая система внешних устройств для ввода изображений (к ним относятся сканеры, видеокамеры, цифровые фотокамеры, графические планшеты).
3. Фотореалистичность (можно получать живописные эффекты, например, туман или дымку, добиваться тончайшей нюансировки цвета, создавать перспективную глубину и нерезкость, размытость и т. д.).
4. Форматы файлов, предназначенные для сохранения точечных изображений, являются стандартными, поэтому не имеет решающего значения, в каком графическом редакторе создано то или иное изображение.
5. Можно использовать в Web-дизайне.

### **Недостатки растровой графики**

1. Объём файла точечной графики однозначно определяется произведением площади изображения на разрешение и на глубину цвета (если они приведены к единой размерности). При этом совершенно неважно, что отображено на фотографии. Если три параметра одинаковы, размер файла будет практически одинаковым.
2. При попытке слегка повернуть на небольшой угол изображение, например, с чёткими тонкими вертикальными линиями, чёткие линии превращаются в чёткие «ступеньки» (это означает, что при любых трансформациях - поворотах, наклонах и т. д. - в точечной графике невозможно обойтись без искажений).
3. Невозможность увеличения изображений для рассмотрения деталей. Поскольку изображение состоит из точек, то увеличение изображения приводит только к тому, что эти точки становятся крупнее. Никаких дополнительных деталей при увеличении растрового изображения рассмотреть не удаётся. Более того, увеличение точек раstra визуальнo искажает иллюстрацию и делает её грубой (пикселизация).

**Растровые форматы:** GIF, BMP, WBMP, PCX, PCD, PSD, FLM, IFF, PXR, PNG, SCT/PICT, PCT, RAW, TIF/TIFF, BMP, JPEG, TGA, FPX, GIF, PhotoCD, MNG, ICO, FLA/SWF.

Дадим краткую характеристику этих форматов.

#### **GIF**

В 1987 году фирма CompuServe представила новый формат для хранения изображений в режиме индексированных цветов. Формат GIF (Graphics Interchange Format) создан крупнейшей сетевой службой CompuServe (ныне подразделение AOL, America OnLine) специально для передачи растровых

изображений в глобальных сетях. В 1989 году формат был модифицирован, и его новая версия получила название gif89a. Gif ориентирован в первую очередь на хранение изображений в режиме индексированных цветов (не более 256), также поддерживает компрессию без потерь LZW. Но главный принцип уменьшения объема картинок в формате gif — это, все-таки, приведение их к меньшему числу цветов. Само собой, что такое пройдет без последствий лишь на картинках с изначально небольшим количеством цветов: рисованной графике, элементах оформления, маленьких надписях (кстати, для хорошего сглаживания надписи классическим шрифтом на однородном фоне достаточно от 7 до 11 цветов в зависимости от кегля).

Используется только по своему первоначальному предназначению - в интернете, поскольку поддерживает только индексированные изображения. Не поддерживает дополнительных каналов, обтравочных контуров, цветовых профилей. Версия GIF 89a позволяет сохранять в одном файле несколько индексированных изображений. Браузеры способны демонстрировать все эти изображения по очереди, получая в результате несложную анимацию. В файле анимации хранятся не только кадры анимации, но и параметры ее демонстрации. GIF анимация в силу своей простоты наиболее распространена в Интернете. Кроме того, один из цветов в палитре индексированного изображения можно объявлять прозрачным. В браузере сквозь участки этого цвета будет виден фон страницы.

Настраиваемая палитра (не более 256 цветов), задаваемая прозрачность одного из цветов, возможность сохранения с чередованием строк (при просмотре сначала выводится каждая 8-я, затем каждая 4-я и т. д. - это позволяет судить об изображении до его полной загрузки). Способен содержать несколько кадров в одном файле с последующей последовательной демонстрацией (т.н. «анимированный GIF»). Уменьшение размера файла достигается удалением из описания палитры неиспользуемых цветов и построчного сжатия данных (записывается количество точек повторяющегося по горизонтали цвета, а не каждая точка с указанием ее цвета). Такой алгоритм дает лучшие результаты для изображений с протяженными по горизонтали однотонными объектами. К сожалению, с 1995 года разработчик GIF компания CompuServe сделала платным любое его использование в программных продуктах (кроме бесплатного программного обеспечения). Это приводит к постепенному вытеснению этого популярного формата из Интернета. Можно сказать, что на поверхности его держит только способность содержать анимацию. Она используется для создания рекламных баннеров.

## **JPEG**

Формат JPEG (Joint Photographic Experts Group) впервые реализовал новый принцип сжатия с потерями информации. Он основан на удалении из изображения той части информации, которая слабо воспринимается человеческим глазом. Лишенное избыточной информации изображение занимает гораздо меньше места, чем исходное. Степень сжатия, а следовательно и количество удаляемой информации, плавно регулируется. Низкие степени сжатия

дают лучшее качество изображения, а высокие могут существенно его ухудшить. Наиболее широко JPEG используется при создании изображений для электронного распространения на компакт-дисках или в интернете. Компактность файлов JPEG делает этот формат незаменимым в тех случаях, когда размер файлов критичен, например, при их передаче по каналам связи. В полиграфии использовать его не рекомендуется, хотя формат допускает хранение цветных профилей и контуров обтравки. JPEG поддерживает полутоновые и полноцветные изображения в моделях RGB и CMYK. Не поддерживаются дополнительные цветовые альфа-каналы. Используется формат JPEG только для хранения фотографических изображений. На рисунках с четкими границами и большими заливочными областями сильно проявляются дефекты сжатия. Особенно характерно проявление грязи вокруг темных линий на светлом фоне и видимых квадратных областей. Последний дефект связан с тем, что алгоритм сжатия обрабатывает изображения квадратными блоками со стороной 8 пикселей.

Миллионы цветов и оттенков, палитра ненастраиваемая, этот формат предназначен для представления сложных фотоизображений. Разновидность progressive JPEG позволяет сохранять изображения с выводом за указанное количество шагов (от 3 до 5 в Photoshop'e) - сначала с маленьким разрешением (плохим качеством), на следующих этапах первичное изображение перерисовывается все более качественной картинкой. Анимация или прозрачный цвет форматом не поддерживаются. Уменьшение размера файла достигается сложным математическим алгоритмом удаления информации - заказываемое качество ниже - коэффициент сжатия больше, файл меньше. Главное - подобрать максимальное сжатие при минимальной потере качества. Кроме коэффициента сжатия еще приходится делать выбор между типами формата - стандартный, оптимизированный или прогрессивный. Наиболее подходящий формат для размещения в интернете полноцветных изображений. Вероятно, до появления мощных алгоритмов сжатия изображения без потери качества останется ведущим форматом для представления фотографий в Web. Плохо, что качество теряется при каждом последующем сохранении.

Существует три подформата jpg: обычный, optimized (файлы несколько меньше, но не поддерживаются старыми программами) и progressive (чересстрочное отображение, аналог interlaced в gif). Некоторые приложения позволяют хранить изображение в jpg в режиме CMYK и даже включать в файл обтравочные контуры. Однако использовать jpg для полиграфических нужд категорически не рекомендуется из-за взаимодействия регулярной структуры блоков 8x8 пикселей, получающихся в результате компрессии, с не менее регулярной структурой типографского раstra, что в итоге приводит к образованию муара. В этом формате лучше сохранять только крупные фотографии с большим количеством плавных цветовых переходов. Кроме того, не стоит сохранять одно и то же изображение в jpg больше одного раза: слишком заметными оказываются деструктивные изменения картинки от повторного использования компрессии.

## **BMP**

Растровый формат BMP (BitMap), созданный Microsoft, ориентирован на применение в операционной системе Windows. Он используется для представления растровых изображений в ресурсах программ. Поддерживаются только изображения в модели RGB с глубиной цвета до 24 бит. Не поддерживаются дополнительные цветовые и альфа-каналы, контуры обтравки, управление цветом. В принципе формат предполагает использование простейшего алгоритма сжатия (Run Length Encoding, RLE) без потерь информации, но этот вариант используется редко из-за потенциальных проблем несовместимости.

## **WBMP**

Последняя версия Photoshop в модуле Save for Web умеет сохранять картинки в формате Wireless Bitmap (WBMP), специально оптимизированном для сотовых телефонов, смартфонов, карманных компьютеров и прочих мобильных устройств. Описание этого формата вместе с языком разметки WML (Wireless Markup Language) включено в спецификацию WAP (Wireless Application Protocol). Кроме Photoshop создавать изображения WBMP способна также Macromedia Fireworks 4 и выше. Формат поддерживает только два цвета, но можно имитировать больше с помощью разброса пикселей (dithering). Теоретически файлы WBMP могут содержать анимацию. Сжатие не поддерживается, что очень удивительно, так как на практике графический файл для WAP не может быть больше 1461 байт (это ограничение связано с небольшим объемом памяти сотовых телефонов). Из-за скромного разрешения дисплеев мобильных устройств безопасный размер файлов ограничен 90x24 пикселями. Помимо вышперечисленных недостатков WBMP еще довольно сыроват: лишь немногие устройства способны отображать графику в этом формате.

## **PCX**

Формат PCX (PC eXchange) - один из первых растровых форматов, созданных фирмой ZSoft для программы PC Paintbrush. Поддерживает монохромные, индексированные и полноцветные изображения модели RGB. Не поддерживаются дополнительные цветовые и альфа-каналы, контуры обтравки, управление цветом. Формат предполагает использование простейшего алгоритма сжатия (Run Length Encoding, RLE) без потерь информации. Ныне имеет преимущественно историческое значение. Свою пальму первенства по примитивизму когда-то взрастил и формат pcx: он почти так же прост внутри, как и bmp. Возможности у этого формата такие же, как и у bmp, только поддержка OS/2 отсутствует. Зато pcx можно посмотреть большинством программ под DOS, в том числе внутренним просмотрщиком Norton Commander.

## **PCD**

Формат PCD (Photo CD) был разработан фирмой Kodak для хранения сканированных фотографических изображений. Сканирование выполняется на

специальной аппаратуре (рабочих станциях Kodak, PIW), а его результат записывается на компакт-диск особого формата, Kodak Photo CD. Его можно просматривать с помощью промышленных видеоплееров и игровых приставок на обычном телевизоре. На практике Photo CD чаще применяются в издательских технологиях как источник изображений. Большинство производителей библиотек фотоснимков используют именно этот формат на своих компакт-дисках. Формат PCD имеет ряд полезных особенностей, делающих эту его область применения преобладающей. Файл PCD содержит изображение сразу в нескольких фиксированных разрешениях. Базовое (Base) разрешение, 512x768 пикселей, используется для просмотра на телевизорах NTSC и PAL. Кроме него имеются пониженные разрешения Base4, Base16 и более высокие 4Base, 16Base и 64Base. Последнее разрешение, 64Base, равное 4096x6144 пикселей, есть только на дисках стандарта Pro Master. Любопытно, что наличие в одном файле шести вариантов одного изображения не увеличивает его размер. Дело в том, что копии высокого разрешения представлены в виде разностей с базовым. Таким образом удастся избежать дублирования графической информации. Изображения на Photo CD представлены в особой цветовой модели YCC, разработанной специалистами Kodak и во многом аналогичной модели Lab. YCC тоже имеет три базовых компонента, яркостный и два хроматических. Поскольку глаз более чувствителен к яркостям, чем к цвету, половина цветовой информации отбрасывается при сканировании: на каждые два пикселя приходится только одно значение хроматических компонентов. Благодаря этому удастся сократить объем графических данных и размер PCD-файла. Для дальнейшего уменьшения размеров файла используется обычная схема сжатия без потерь качества LZW. Существуют несколько форматов Photo CD. Формат Master Photo CD содержит изображения, сканированные с обычной фотопленки формата 35 мм. Максимальное разрешение для этого типа 16 Base. Профессиональным фотографам адресован формат Master Pro Photo CD, для которого используется пленка большего формата (120 мм и 4x5 дюймов). Для полиграфических приложений предназначен формат Print Photo CD. Оригинал сканируется профессиональными сканерами (Crosfield, Linotype, Scitex) и сохраняется с несжатым разрешением 64 Base. Формат Catalog Photo CD позволяет разместить на одном диске до 4500 изображений с базовым разрешением. И наконец, на мультимедийные приложения ориентирован формат Portfolio PhotoCD. На компакт-диске такого формата можно разместить до 800 изображений, а также звук, интерактивные сценарии и т. п.

### **TIF, TIFF**

Формат TIFF (Tagged Image File Format) создан объединенными силами таких гигантов, как Aldus, Microsoft и Next специально для хранения сканированных изображений. Исключительная гибкость формата сделала его действительно универсальным. TIFF - один из самых древних форматов в мире микрокомпьютеров, на сегодняшний день он является самым гибким, универсальным и активно развивающимся. В нем можно хранить графику в любом режиме: от битового и индексированных цветов до Lab, CMYK и RGB (кроме дуп-

лексов и многоканальных документов). Хотя с момента его создания прошло уже много времени, TIFF до сих пор является основным форматом, используемым для хранения сканированных изображений и размещения их в издательских системах и программах иллюстрирования. Версии формата существуют на всех компьютерных платформах, что делает его исключительно удобным для переноса растровых изображений между ними. TIFF поддерживает монохромные, индексированные, полутоновые и полноцветные изображения в моделях RGB и CMYK с 8- и 16-битными каналами. Он позволяет хранить обтравочные контуры, калибровочную информацию, параметры печати. Допускается использование любого количества дополнительных альфа-каналов. Дополнительные цветовые каналы не поддерживаются. Большим достоинством формата остается поддержка практически любого алгоритма сжатия. Наиболее распространенным является сжатие без потерь информации по алгоритму LZW (Lempel Ziv Welch), обеспечивающему очень высокую степень компрессии. Кстати, этот же алгоритм используется многочисленными программами сжатия общего назначения, поддерживающими формат ZIP.

### **PSD**

Формат PSD (PhotoShop Document) - это собственный формат программы Adobe Photoshop. Единственный формат, поддерживающий все возможности программы. Предпочтителен для хранения промежуточных результатов редактирования изображений, так как сохраняет их послойную структуру. Все последние версии продуктов фирмы Adobe Systems поддерживают этот формат и позволяют импортировать файлы Photoshop непосредственно. К недостаткам формата PSD можно отнести недостаточную совместимость с другими распространенными приложениями и отсутствие возможности сжатия.

Поддерживаются все цветовые модели и любая глубина цвета от бело-черного до true color, сжатие без потерь. Начиная с версии 3.0 Adobe добавила поддержку слоев и контуров, поэтому формат версии 2.5 и ранее выделяется в отдельный подформат. Для совместимости с ним в более поздних версиях Photoshop имеется возможность включить режим добавления в файл одного базового слоя, в котором слиты все слои. Такие файлы свободно читаются большинством популярных просмотрщиков, импортируются в другие графические редакторы и программы для 3D моделирования.

### **FLM**

FLM (Filmstrip) - собственный формат Adobe Premier, программы редактирования видеоинформации и создания презентаций.

### **IFF**

Формат IFF (Amiga Interchange File Format) используется на компьютерах Commodore Amiga с программно-аппаратным комплексом Video Toaster. Он ориентирован на создание и обработку высококачественных видеоматериалов в реальном времени. Поддерживается также некоторыми программами рисования на платформе Windows, например Deluxe Paint фирмы Electronic

Arts. Формат IFF поддерживает все типы изображений за исключением многоканальных и полноцветных CMYK. Обтравочные контуры, цветовые профили и альфа-каналы не поддерживаются.

### **PXR**

Формат PXR (Pixar) предназначен для обмена со специализированными графическими станциями Pixar, ориентированными на трехмерное моделирование и анимацию. Поддерживаются только полутоновые и полноцветные RGB изображения с единственным альфа каналом.

### **PNG**

На сегодня самый прогрессивный формат графики для интернета - это png (Portable Network Graphics, читается «пинг»). Этот луч света и уникальное решение множества проблем позволяет создавать «зоны прозрачности», как .gif, но в довесок может быть и полупрозрачным (сквозь него может просвечивать фон). Он был, по сути, выпадом независимых групп и консорциумов в сторону компании Ulead, которая в 1995 году присвоила себе народный алгоритм сжатия без потерь LZW . Вместо последнего в формате png используется алгоритм Deflate, дающий, кстати, несколько лучшие результаты, чем LZW . Изначально призванный заменить морально устаревший gif на искусственных ландшафтах Сети, png предлагает целый ряд новых возможностей, недостаток которых в gif не раз делал его объектом бессильных ругательств.

Это достаточно «молодой» формат для Web-графики, конкурирующий с GIF. Все последние версии браузеров поддерживают его без специальных подключаемых модулей. Формат поддерживает полутоновые и полноцветные RGB-изображения с единственным альфа-каналом, а также индексированные и монохромные изображения без альфа-каналов. Альфа-канал служит маской прозрачности. Таким образом, формат PNG - единственный из распространенных в Интернете форматов, позволяющий получать полноцветные изображения с прозрачным фоном. В формате PNG использован мощный алгоритм сжатия без потерь информации, основанный на популярном LZW-сжатии. Будучи ориентированным на Web, формат PNG не поддерживает многоканальных изображений, цветовых профилей и контуров обтравки.

Существует два подформата: PNG8 и PNG24, цифры означают максимальную глубину цвета, возможную в подформате. В PNG24 наконец-то была реализована поддержка 256 градаций прозрачности за счет дополнительного альфа-канала с 256 градациями серого. С помощью этой функции, например, полупрозрачный логотип может выглядеть одинаково на абсолютно любом фоне. К тому же формат png нашпигован такими полезными возможностями, как двумерный interlacing (т. е. изображение проявляется постепенно не только по строкам, но и по столбцам) и встроенная гамма-коррекция, позволяющая сохранять изображения, яркость которых одинакова как на PC, так и на компьютерах Mac, Sun и Silicon Graphics. Ни одна из полезных функций не поддерживается ни одним из существующих браузеров. PNG8 - малораспространен из-за слабой рекламы, создавался специально для интернета как замена



первых двух форматов и благодаря патентной политике CompuServe постепенно вытесняет GIF. Позволяет выбирать палитру сохранения - серые полутона, 256 цветов, true color (истинные цвета). В зависимости от свойств изображения действительно иногда предпочтительнее GIF-а. Позволяет использовать «прозрачный» цвет, но, в отличие от GIF-а таких цветов может быть до 256. В отличие от GIF сжатие без потери качества производится и по горизонтали, и по вертикали (алгоритм собственный, параметры тоже ненастраиваемые). Не поддерживает анимацию.

Почему же такой замечательный и удобный формат не получил распространения? Дело в том, что фирма Microsoft сочла этот формат неперспективным на основании того, что разработан он не ими.

### **SCT**

Формат SCT (Scitex Continuous Tone) используется сканерами, фотонаборными автоматами и графическими станциями Scitex для получения высококачественной полиграфической продукции. Особый формат используется патентованным растеризатором Scitex. Он поддерживает полутоновые и полноцветные изображения в моделях RGB и CMYK без альфа-каналов. Обтравочные контуры и цветовые профили не поддерживаются.

Scitex используется исключительно на этапе растривания смеси из векторных и растровых данных в одну битовую карту, предназначенную для high-end фотонаборных автоматов фирмы Scitex - она то и сохраняется в этом формате. Можно не доверять растривание файла сервисному бюро, а самому сохранить макет в формате Scitex. Он не поддерживает никаких алгоритмов сжатия.

### **PCT/PICT**

Pict (Macintosh QuickDraw Picture Format) - это внутренний формат операционной системы Mac, аналог bmp. Он способен нести в себе растровую и векторную информацию, текст и даже звук. Такая потрясающая гибкость формата лишней раз подтверждает эффективность использования Mac при работе с мультимедиа. Изображение может храниться как в RGB, так и в CMYK, причем глубина цвета варьируется от индексированных цветов до true color; реализован алгоритм компрессии без потерь RLE. Формат pict открывается всеми приложениями, разработанными для Mac (QuickTime, Photoshop, etc.)

### **RAW**

Замечательный контраст с предыдущим форматом составляет формат raw. Он не поддерживает ничего. То есть совсем. Не хранятся даже данные о количестве каналов, глубине цвета и разрешении, так что во время открытия вам придется вводить эти параметры вручную, по памяти. Изображение хранится просто как поток пикселей с фиксированным заголовком, куда можно впоследствии поместить любую текстовую информацию. Кстати, размер этого заголовка в байтах вам тоже придется указывать при открытии картинки в этом формате. Это сделано для обеспечения мультиплатформенности и совме-

стимости со всеми программами. Однако далеко не каждый графический редактор или просмотрщик поддерживает raw.

### **TGA**

Довольно старый формат TGA (targa) создан специально для работы с графическим акселератором TrueVision. Этот акселератор широко используется приложениями на платформе DOS. Формат поддерживает 24-битное и 32-битные RGB изображения с одним альфа-каналом, а также полутоновые, индексированные и 16-битные RGB изображения без альфа-каналов. Обтравочные контуры и цветовые профили не поддерживаются. Также пользуется уважением среди программ DOS формат targa (Truevision Targa Image File). Он поддерживает глубину цвета от 8 до 32 бит на пиксель и использует алгоритм компрессии без потерь RLE. Файлы формата targa часто применялись DOS версией 3DStudio Max для хранения текстур.

### **FPX**

Еще один формат, не ставший популярным ввиду слабой маркетинговой поддержки - это FlashPix. Он был разработан фирмой Kodak, известной по формату PhotoCD своими попытками загнать в один файл несколько копий одного и того же изображения с разными разрешениями. FlashPix не стал исключением из фирменного правила и тоже поддерживает несколько копий с разным разрешением в одном файле. Веб-дизайнер, никогда не сталкивавшийся с файлами полиграфического качества, возможно спросит о смысле такого расточительного расходования дискового пространства. Он есть. Дело в том, что в полиграфии нередко работают с изображениями, занимающими десятки и даже сотни мегабайт. Их приведение к нужному размеру занимает гораздо больше времени, чем просто считывание копии с нужным разрешением, а размер файлов в предпечатной подготовке роли не играет. FlashPix также обладает встроенной системой защиты изображений с помощью водяных знаков. Формат достаточно редкий, и немногие программы умеют с ним работать.

### **PXR**

Для рядового пользователя Pixar - всего лишь музейная редкость. Это и понятно: он применяется исключительно на high-end графических станциях Pixar, предназначенных для профессиональной трехмерной анимации. Его возможности невелики: отсутствие компрессии, поддержка лишь модели RGB и градаций серого и одного альфа-канала.

### **ICO**

ICO - формат мелких картинок (иконок) в WWW. Картинки используются браузерами для маркировки Web-проектов в строке URL и в избранном. Поддерживается и используется программками для создания иконок типа IconXP.

## **FLA**

FLA - внутренний формат программы для создания интерактивной анимации Flash

## **SWF**

SWF - формат публикации Flash для отображения на разных платформах.

**Векторные форматы:** WMF, EMF, CGM, EPS, WPG, AutoCAD, DXF, DWG, CDR, AI, PCT, FLA/SWF

## **EPS (Encapsulated PostScript)**

Благодаря своей надежности, совместимости со многими программами и платформами и большому числу настраиваемых параметров формат eps (Encapsulated PostScript) является выбором большинства профессионалов в области полиграфии. Он предназначен сугубо для переноса готовых изображений в программы верстки, поддерживает цветовые модели CMYK, RGB, дуплексы и содержит готовые команды устройству вывода. В eps можно сохранить информацию о треппинге, типографском растре, внедренных шрифтах и обтравочных контурах. Данные хранятся тремя способами: ASCII (медленный, но наиболее совместимый), Binary (быстрый и компактный), JPEG (быстрый, но с потерями качества и плохой совместимостью). При сохранении в eps можно указать формат и глубину цвета эскиза, который для ускорения работы будет выводиться на экран в программах верстки вместо большого оригинала.

## **PCT**

Pict (Macintosh QuickDraw Picture Format) - это внутренний формат операционной системы Mac, аналог bmp. Он способен нести в себе растровую и векторную информацию, текст и даже звук. Такая потрясающая гибкость формата лишней раз подтверждает эффективность использования Mac при работе с мультимедиа. Изображение может храниться как в RGB, так и в CMYK, причем глубина цвета варьируется от индексированных цветов до true color; реализован алгоритм компрессии без потерь RLE. Формат pict открывается всеми приложениями, разработанными для Mac (QuickTime, Photoshop, etc.)

## **FLA**

FLA - внутренний формат программы для создания интерактивной анимации Flash

## **SWF**

SWF - формат публикации Flash для отображения на разных платформах.

Наиболее распространенные форматы файлов для печати: TIFF, EPS для web-дизайна: PNG, GIF, SWF, JPG.

### 3.4. Графические редакторы

В настоящее время существует достаточно большое количество редакторов графических изображений. Перечислим наиболее распространенные из них.

#### Редакторы векторной графики и анимации:

Macromedia Freehand, Macromedia Flash, Adobe Illustrator, Adobe Streamline, CorelDRAW, Corel Xara и др.

#### Редакторы растровой графики и анимации:

Adobe Photoshop, Adobe ImageReady, PaintShop Pro, Animation Shop, PhotoPaint, Painter, Image 2000, LView Pro, Microsoft PhotoDRAW, Microsoft Photo Editor, Microsoft Paint и др.

#### Редакторы 3D графики и анимации:

3D Studio MAX, Xara 3D, CorelDream 3D, trueSpace, Bryce, World Construction Set, Piasma, 3D VIZ, Organica, Maya и др.

### 3.5. Графика в системе MATLAB

MATLAB имеет широкие возможности для графического изображения векторов и матриц, а также для создания комментариев и печати графики. Эта глава описывает несколько наиболее важных графических функций и дает примеры их применения.

#### 3.5.1. Создание графиков

Функция **plot** имеет различные формы, связанные с входными параметрами, например **plot(y)** создает кусочно-линейный график зависимости элементов  $y$  от их индексов. Если задать два вектора  $x$  и  $y$  в качестве аргументов, **plot(x,y)** создаст график зависимости  $y$  от  $x$ .

Например, для построения графика значений функции **sin** от нуля до  $2\pi$ , сделаем следующее:

```
t = 0:pi/100:2*pi;  
y = sin(t);  
plot(t,y)  
grid on
```

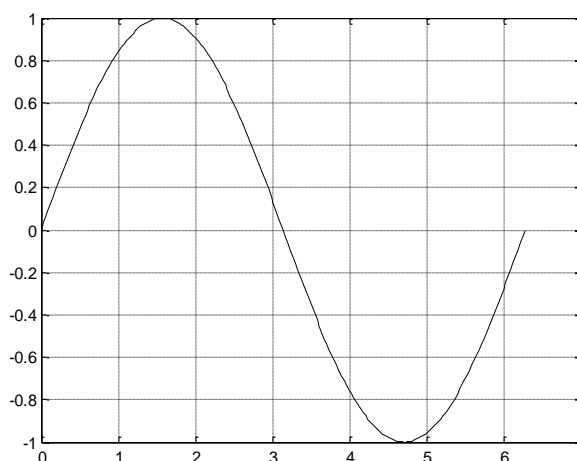


Рис. 3.2. Пример графика  $y = \sin(t)$ , выведенного с помощью функции **plot**

Вызов функции **plot** с многочисленными парами  $x$ - $y$  создает несколько графиков. MATLAB автоматически присваивает каждому графику свой цвет

(исключая случаи, когда это делает пользователь), что позволяет различать заданные наборы данных. Например, следующие три строки, продолжающие предыдущую программу, отображают график близких функций, и каждой кривой соответствует свой цвет, что видно на экране компьютера и не отражается в черно-белой печати.

```
y2 = sin(t-.25);
y3 = sin(t-.5);
plot( t, y, t, y2, t, y3)
```

Возможно изменение цвета, стиля линий и маркеров, таких, как знаки плюс или кружки, следующим образом

```
plot(x, y, 'цвет_стиль_маркер')
```

где цвет\_стиль\_маркер это 1-, 2-, 3-х символьная строка (заклученная в одинарные кавычки), составленная из типов цвета, стиля линий и маркеров:

- символы, относящие к цвету: 'c', 'm', 'y', 'r', 'g', 'b', 'w' и 'k', они обозначают голубой (cyan), пурпурный (magenta), желтый (yellow), красный (red), зеленый (green), синий (blue), белый (white) и черный (black) цвета соответственно;
- символы, относящиеся к типу линий: '-' для сплошной, '—' для разрывной, ':' для пунктирной, '-.' для штрихпунктирной линий и 'none' для её отсутствия;
- наиболее часто встречающиеся маркеры '+', 'o', '\*' и 'x'.

Например, выражение

```
plot(x,y,'y:+')
```

строит желтый пунктирный график и помещает маркеры '+' в каждую точку данных. Если вы определяете только тип маркера, но не определяете тип стиля линий, то MATLAB выведет только маркеры.

## Окна изображений

Функция **plot** автоматически открывает новое окно изображения, если до этого его не было на экране. Если же оно существует, то **plot** использует его по умолчанию. Для открытия нового окна и выбора его по умолчанию используется функция

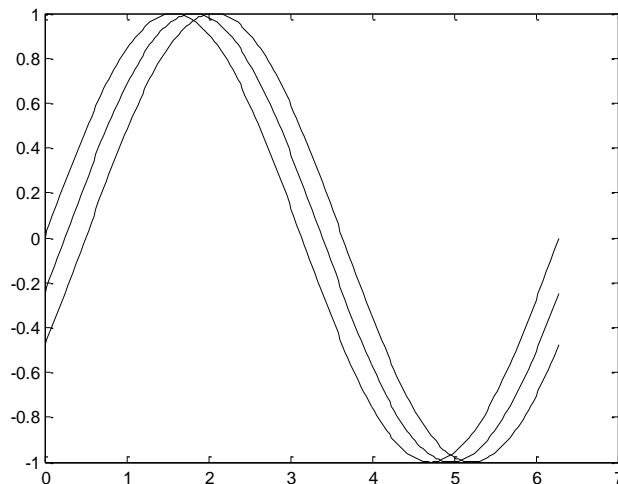


Рис. 3.3. Пример трех графиков, выведенных с помощью функции **plot**

figure

Для того, чтобы сделать существующее окно текущим, указывается номер окна

figure(n)

где n - это номер в заголовке окна. В этом случае результаты всех последующих команд будут выводиться в это окно.

### Добавление кривых на существующий график

Команда **hold** позволяет добавлять кривые на существующий график. Если набрать

hold on

MATLAB не стирает существующий график, а добавляет в него новые данные, изменяя оси, если это необходимо. Например, следующий элемент кода вначале создает контурные линии функции **peaks**, а затем накладывает псевдоцветной график той же функции

```
[x,y,z] = peaks;  
contour(x,y,z,20,'k')
```

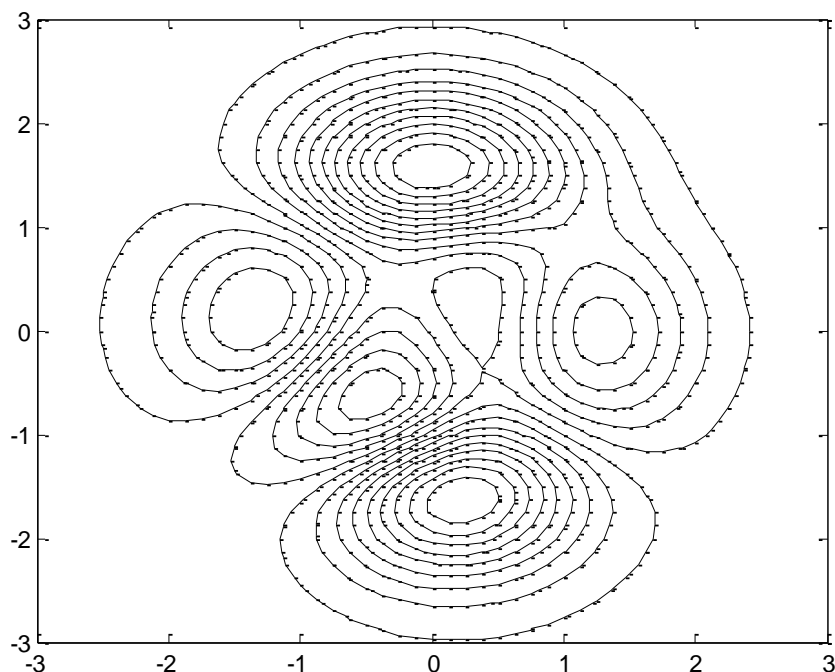


Рис. 3.4. Результат построения контурного графика

Здесь **peaks** представляет пример функции двух переменных, образующейся при преобразовании и изменении масштаба гауссова распределения.

Результатом выполнения будет построение изображения изолиний, представленного на рис. 3.4. Оно образуется сечением горизонтальными плоскостями 5-вершинной поверхности.

Продолжение этого фрагмента программы производит заполнение цветом и нанесение цветowych теней на том же чертеже.


```
hold on  
pcolor(x,y,z)  
shading interp
```

Команда **hold on** является причиной того, что график **pcolor** комбинируется с графиком **contour** в одном окне. Из-за черно-белой печати данные построения здесь не приводятся, чтобы увидеть результат, их нужно построить непосредственно в системе MATLAB.

Изображение поверхности может быть получено с помощью функции **surface** следующим образом (продолжается предыдущий фрагмент программы):

```
figure(2)  
surface(x,y,z)
```

В первой строчке открывается новое окно, а во второй строится поверхность.

В результате будет построено изображение поверхности при взгляде сверху. Его можно развернуть в пространстве с помощью нажатия кнопки  на панели управления изображениями в окне, куда выводится график, последующим захватом левой кнопкой мыши и вращением. На рис. 3.5 показано изображение поверхности, для которой был построен контурный график (карта изолиний), изображенный на рис. 3.4.

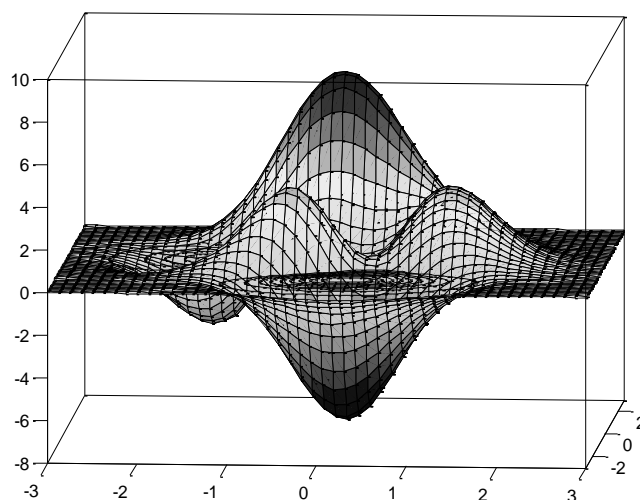


Рис. 3.5. Изображение демонстрационной поверхности **peaks**, построенное функцией **surface**

### 3.5.2 Подграфики

Функция **subplot** позволяет выводить множество графиков в одном окне или распечатывать их на одном листе бумаги.

```
subplot(m,n,p)
```

разбивает окно изображений на матрицу  $m \times n$  подграфиков и выбирает  $p$ -ый подграфик текущим. Графики нумеруются, начиная с первого левого в верхней строке, потом во второй и т. д. Например, для того, чтобы представить графические данные в четырех разных подобластях окна, необходимо выполнить следующее

```
t = 0:pi/10:2*pi;  
[X,Y,Z] = cylinder(4*cos(t));  
subplot(2,2,1)  
mesh(X)  
subplot(2,2,2); mesh(Y)  
subplot(2,2,3); mesh(Z)  
subplot(2,2,4); mesh(X,Y,Z)
```

Результаты выполнения этого фрагмента представлены на рис. 3.6

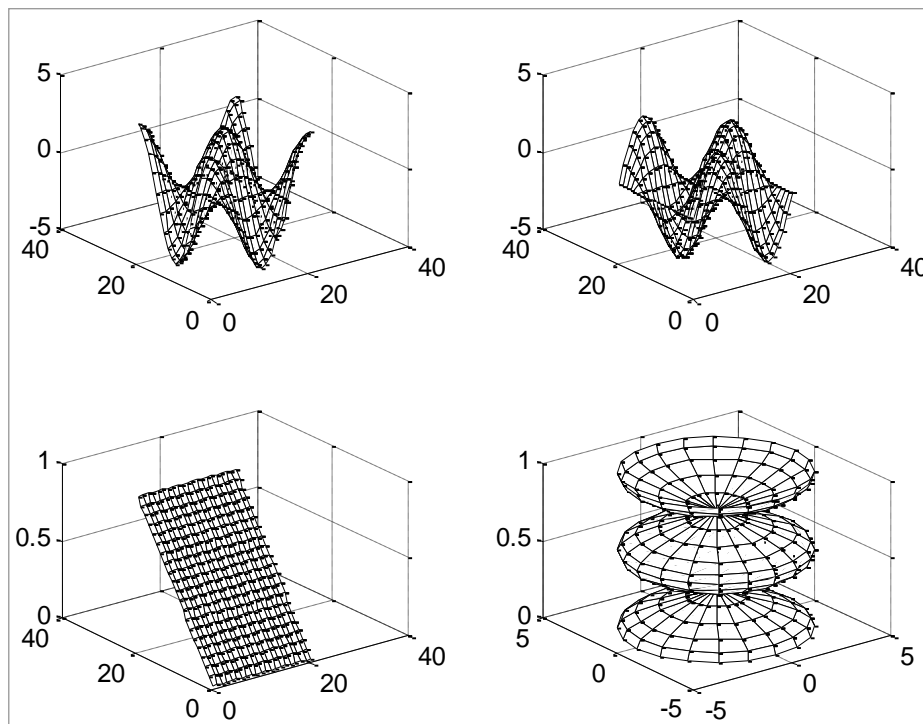


Рис. 3.6. Представление результатов расчетов в нескольких окнах



### 3.5.3 Управление осями

Функция **axis** имеет несколько возможностей для настройки масштаба, ориентации и коэффициента сжатия.

Обычно **MATLAB** автоматически находит максимальное и минимальное значение и выбирает соответствующий масштаб и маркирование осей. Функция **axis** заменяет значения по умолчанию предельными значениями, вводимыми пользователем. Формат функции

```
axis([xmin xmax ymin ymax])
```

где *xmin*, *xmax*, *ymin*, *ymax* – минимальные и максимальные значения по осям *x* и *y* соответственно.

В функции **axis** можно также использовать ключевые слова для управления внешним видом осей. Например

```
axis square
```

создает оси *x* и *y* равной длины, а

```
axis equal
```

создает отдельные отметки приращений для *x* и *y* осей одинаковой длины. Так функция

```
plot(exp(i*t))
```

следующая либо за **axis square**, либо за **axis equal** превращает овал в правильный круг.

```
axis auto
```

возвращает значения по умолчанию и переходит в автоматический режим.

```
axis on
```

включает обозначения осей и метки промежуточных делений, а

```
axis off
```

выключает обозначения осей и метки промежуточных делений.

Функция

```
grid off
```

выключает сетку координат, а

```
grid on
```

включает её заново.

### 3.5.4 Подписи к осям и заголовки

Функции **xlabel**, **ylabel**, **zlabel** добавляют подписи к соответствующим осям, функция **title** добавляет заголовок в верхнюю часть окна, а функция **text** вставляет текст в любое место графика. Использование TEX-представления позволяет применять греческие буквы, математические символы и различные шрифты. Следующий пример демонстрирует эту возможность.

```
t = -pi:pi/100:pi;  
y = sin(t) ;  
plot(t,y)  
axis([-pi pi -1 1])  
xlabel( '-\pi \leq t \leq \pi ' )  
ylabel( ' sin(t) ' )  
title( ' График функции sin ' )  
text(-1, -1/3, '\it{Комментирующее пояснение} ' )
```

Результат показан на рис. 3.7.

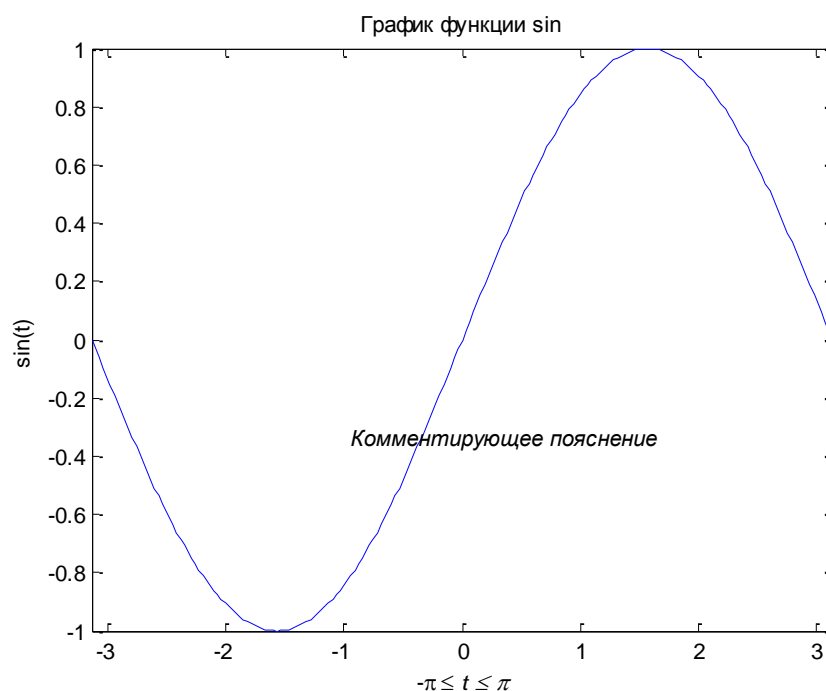


Рис. 3.7. Вывод графика с обозначением осей и поясняющими надписями

### 3.5.5 Функции `mesh` и `surface`

MATLAB определяет поверхность как координаты  $z$  точек над координатной сеткой плоскости  $x$ - $y$ , используя прямые линии для соединения соседних точек. Функции `mesh` и `surface` отображают поверхность в трех измерениях. При этом `mesh` создает каркасную поверхность, где цветные линии соединяют только заданные точки, а функция `surface` вместе с линиями отображает в цвете и саму поверхность.

### 3.5.6. Визуализация функций двух переменных

Для отображения функции двух переменных,  $z = f(x, y)$ , создаются матрицы  $X$  и  $Y$ , состоящие из повторяющихся строк и столбцов соответственно, перед использованием функции. Затем используют эти матрицы для вычисления и отображения функции. Функция `meshgrid` преобразует область определения, заданную через один вектор или два вектора  $x$  и  $y$ , в матрицы  $X$  и  $Y$  для использования при вычислении функции двух переменных. Строки матрицы  $X$  дублируют вектор  $x$ , а столбцы  $Y$  - вектор  $y$ .

Для вычисления двумерной функции `sinc`, `sin(r)/r`, в области  $x$ - $y$  поступают следующим образом:

```
[X, Y] = meshgrid(-8:.5:8);  
R = sqrt(X.^2+Y.^2)+eps;  
Z = sin(R)./R; mesh (X, Y, Z)
```

Результат построений представлен на рис. 3.8.

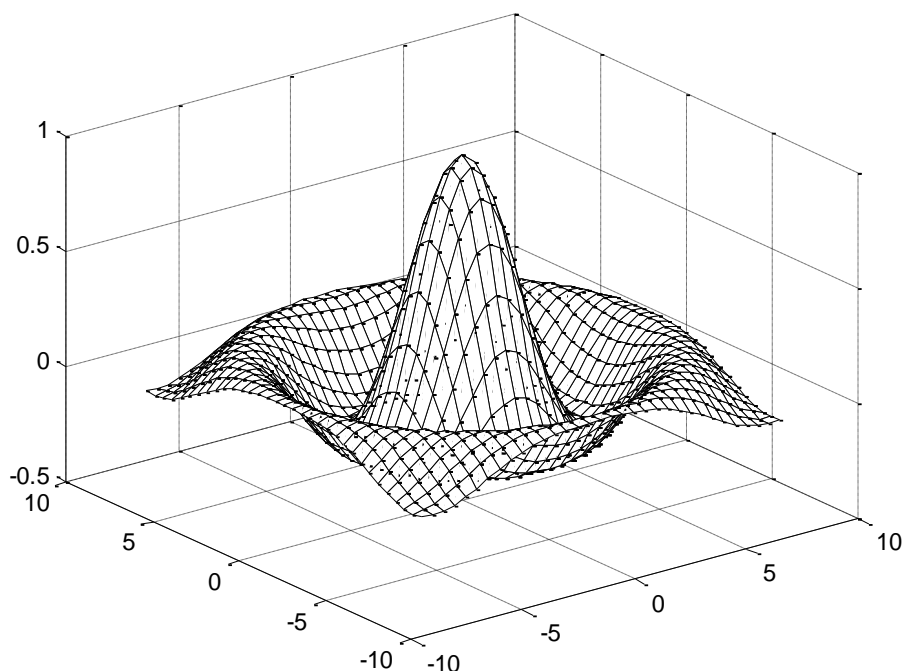


Рис. 3.8. Вывод изображения поверхности

В этом примере  $R$  - это расстояние от начала координат, которому соответствует центр матрицы. Добавление **eps** позволяет избежать неопределенности  $0/0$  в начале координат.

## Изображения

Двумерные массивы могут отображаться как изображения, где элементы массива определяют их яркость и цвет. Например

```
load durer;  
whos
```

покажет, что файл **durer.mat** в директории **demo** состоит из матрицы размером 648 на 509 (матрицы  $X$ ) и матрицы размером 128 на 3 (матрицы  $map$ ). Элементы матрицы  $X$  - это целые числа от 1 до 128, которые служат индикаторами в цветном отображении. Следующие строки

```
image(X)  
colormap(map)  
axis image
```

воспроизводят гравюру Дюрера. Высокое разрешение магического квадрата, находящегося в правом верхнем углу, доступно в другом файле. Для повышения разрешения следует набрать

```
load detail
```

и использовать стрелку «вверх» на клавиатуре для повторного запуска вышеуказанных команд **image**, **colormap** и **axis**. При этом

```
colormap(hot)
```

добавит особую цветовую гамму.

### 3.5.7. Печать графики

Опция **Print** в меню **File** и команда **print** печатают графику MATLAB. Меню **Print** вызывает диалоговое окно, которое позволяет выбирать общие стандартные варианты печати. Команда **print** обеспечивает большую гибкость при выводе выходных данных и позволяет контролировать печать из M-файлов. Результат может быть послан прямо на принтер, выбранный по умолчанию, или сохранен в заданном файле. Возможно широкое варьирование формата выходных данных, включая использование **PostScript**.

Например, следующая команда сохранит текущее окно изображения как цветное **PostScript Level 2 Encapsulated** в файле **magicsquare.eps**:

```
print -depsc2 magicsquare.eps
```

Важно знать возможности принтера перед использованием команды **print**. Например, файлы **Postscript Level 2** обычно меньше и воспроизводятся намного быстрее, чем **Postscript Level 1**. Однако не все Postscript принтеры поддерживают **Level 2**, таким образом, необходимо узнать, что может обрабатывать используемое устройство вывода. MATLAB использует дифференцированный подход для вывода графики и текста, даже для черно-белых устройств.

## КОНТРОЛЬНЫЕ ВОПРОСЫ К ГЛАВЕ 3

1. Что такое цифровое изображение?
2. Что такое пиксель?
3. В каких видах могут быть представлены цифровые изображения?
3. Какими параметрами характеризуются растровые изображения?
4. Назовите наиболее распространенные модели цифровых изображений, дайте их характеристику.
5. Что такое глубина цвета, каким параметром она характеризуется?
6. Перечислите наиболее употребительные форматы файлов изображений.
7. В чем особенность деструктивного сжатия и как оно влияет на качество изображений?
8. Векторная графика, принципы организации, достоинства и недостатки.
9. Растровая графика, принципы организации, достоинства и недостатки.
10. Что такое сплайн, кривые Безье, как задается их форма?
11. Чем отличаются друг от друга при кодировке черно-белое (бинарное), серое и цветное изображения?
12. Дайте характеристику наиболее распространенных растровых и векторных форматов представления изображений.
13. Перечислите наиболее распространенные программы для работы с графическими изображениями.
14. Перечислите основные функции системы MATLAB, используемые для создания и работы с графиками зависимостей одних величин от других.
15. Каким образом можно устанавливать различные режимы построения графиков функций (цвет, вид линии, обозначения точек, осей и т. д.)?
16. Можно ли построить несколько кривых в одних и тех же координатных осях, если да, то каким образом?
17. Как выводить графики и изображения в новые окна, а также множество графиков в одном окне?
18. Как добавлять кривые на уже существующий график?
19. Как строить контурные графики и изображения поверхностей?
20. Как осуществлять настройки масштаба, ориентации и коэффициента сжатия, а также выводить подписи к осям и заголовки на графиках?
21. Какие возможности существуют в системе MATLAB для визуализации функции двух переменных?
22. Каким образом можно из программы распечатать изображение на принтере?

## 4. ПОЛУЧЕНИЕ И ОБРАБОТКА ИЗОБРАЖЕНИЙ

### 4.1. Получение изображений

Помимо графического вывода результатов расчетов в научных исследованиях часто осуществляется работа с изображениями, введенными в компьютер с помощью аппаратных устройств.

Изображение в компьютер может быть введено различными способами. Здесь мы не будем рассматривать ввод видео, представляющего собой последовательность отдельных кадров, зарегистрированных через определенные промежутки времени, а ограничимся только одиночными изображениями. В качестве примера рассмотрим изображение аншлифа образца горной породы. Такие изображения могут быть получены с помощью различных устройств: фотоаппарата, видеокамеры, сканера, микроскопа с цифровым преобразователем и др. (рис. 4.1). В цифровых фото- и видеокамерах имеется внутренняя память, поэтому изображения преобразуются в цифровой вид и записываются в виде файлов на внутренний носитель, после чего эти файлы изображений переносятся на жесткий диск компьютера и могут дальше обрабатываться. Такие устройства, как сканеры и вебкамеры не имеют внутренней памяти и в процессе получения электронных изображений они должны передаваться в компьютер и уже там преобразовываться и записываться в виде файлов. Связь этих устройств, как правило, осуществляется через порт **USB** (Universal Serial Bus), который имеет ряд преимуществ по сравнению с другими и используется в настоящее время наиболее часто. Упрощенная схема подключения преобразователей изображений в электрический сигнал представлена на рис. 4.1.

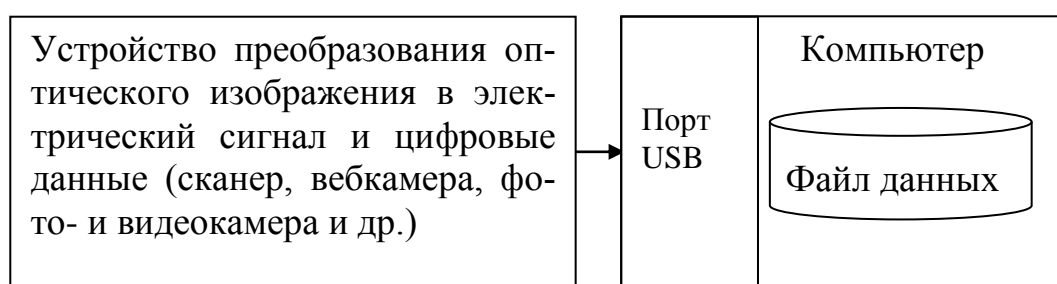


Рис. 4.1. Обобщенная схема получения цифровых изображений

Для того, чтобы отсканировать поверхность горной породы, следует изготовить аншлиф, т. е. ровную и заполированную поверхность. Для сканирования могут быть использованы стандартные сканеры. При выборе сканера следует обращать внимание на его основную характеристику – разрешающую способность. Она не должна быть хуже, чем размеры элементов изображения, которые следует сканировать и обрабатывать. Это означает, что расстояние между отдельными элементами изображения, пикселями, не должно быть больше минимального размера элементов.

Так, например, для распознавания текстов, а также других офисных работ стандартной считается разрешающая способность 300 dpi (точек/дюйм). Для перевода этой величины в метрическую систему следует указанную цифру поделить на 25,4 мм. Это дает величину  $300/25,4 = 11.811$  точек на мм. Под термином «разрешающая способность» понимают также и обратную величину, т. е. минимальное расстояние между точками, которое можно различить. Именно на эту величину нужно ориентироваться при выборе сканера. Так, например, размеры зерен магнетита находятся в диапазоне от 0,01 мм до 0,1 мм. Если использовать сканер с разрешающей способностью 300 dpi, то он будет способен различить элементы изображения размером  $25,4 \text{ мм} / 300 = 0,0847$  мм, большинство из указанных зерен не найдет четкого отражения на изображении и будет потеряно. Для того чтобы отсканировать такое изображение без потерь, следует выбрать сканер с разрешающей способностью  $1 / 0,01 \text{ мм} \cdot 25,4 = 2540 \text{ dpi}$ . Современные планшетные сканеры позволяют отсканировать изображение с разрешающей способностью 4800 dpi, и с помощью такого сканера задача может быть решена.

В нашем случае для примера будем оперировать с изображением аншлифа руды, содержащей железистый кварцит, полученного с помощью сканера с разрешающей способностью 1600 dpi, что позволит различать элементы размером  $25,4 / 1600 = 0,0159$  мм, что почти удовлетворяет указанным требованиям.

При выборе сканера следует учесть, что существует два типа параметров:

- прямая разрешающая способность, достигаемая непосредственно путем сканирования;
- разрешающая способность за счет интерполяции, достигаемая за счет первичной обработки изображения непосредственно в сканере.

Для решения указанной выше задачи необходимо ориентироваться именно на первый параметр, т. е. на разрешающую способность, достигаемую непосредственно за счет механических, оптических и электрических свойств сканера без дополнительной программной интерполяции.

Исходные электронные изображения могут быть получены и с помощью цифровой фотокамеры. Если речь идет о мелких объектах, то следует использовать камеру с режимом макросъемки.

Еще одним из приборов, с помощью которых возможно получить электронные изображения поверхностей горных пород, является оптический микроскоп с цифровой фотокамерой. Для ввода таких изображений в компьютер используется специальное программное обеспечение.

## **4.2. Виды обработки изображений**

После того, как изображения получены и записаны в виде файлов, их часто приходится обрабатывать, т. е. производить определенные операции, которые своей целью имеют либо выделение каких-либо особенностей изображе-



ний, например, граней или границ, либо улучшение изображений, например, улучшение контраста, либо получение каких-либо характеристик, например, размеров элементов.

К наиболее часто применяемым операциям с изображениями относятся следующие:

- пространственные преобразования изображений;
- морфологические операции;
- операции с блоками и соседними элементами;
- линейная фильтрация и разработка фильтров;
- различные трансформации;
- анализ и улучшение изображений;
- запись изображений;
- улучшение резкости;
- операции с заданными областями.

Остановимся на основных видах работы с изображениями, считая, что они уже получены тем или иным способом.

### 4.3. Первичные преобразования изображений

Вначале рассмотрим, как читать изображения и размещать их в рабочем пространстве MATLAB, определять параметры изображений, трансформировать цветные изображения в черно-белые, корректировать контрастность, записывать на диск.

#### Чтение и запись изображений на компьютерах

Как было сказано выше, для работы с изображениями, как и с любыми данными, необходимо владеть их записью и чтением. Рассмотрим эти операции на конкретных примерах.

Поскольку для работы с изображениями требуется достаточно большой объем оперативной памяти, а также для того, чтобы работе не мешали ранее определенные переменные, с помощью функции **close all** рекомендуется закрыть окна, в которые до этого могли быть выведены графики или изображения MATLAB и произвести очистку рабочей области **clear** следующим образом

```
close all;  
clear;
```

Чтение изображений из файла осуществляется с помощью команды **imread**. Например, строки

```
img='magpart.jpg';  
I=imread(img);
```

присваивают переменной **img** значение '**magpart.jpg**' и осуществляют чтение данных из файла с таким названием. При этом файл **magpart.jpg** должен находиться в рабочем каталоге.

Остановимся более подробно на формате функции **imread**. С ее помощью осуществляется чтение серого или цветного изображения. Формат **a = imread(filename,fmt)**, где **filename** – строковая переменная, содержащая название файла, **fmt** – строковая переменная, определяющая его расширение, указывать его здесь необязательно, достаточно записать в переменной **filename**, как это сделано в примере выше. Если файл находится не в рабочем каталоге (**Current Folder**), то необходимо указывать полный путь. В переменной **I** содержатся данные об изображении. Если изображение серое, то **I** представляет собой матрицу размером  $M \times N$ , если же цветное, то матрицу размером  $M \times N \times 3$ , а для файлов, использующих систему цвета CMYK (например, TIFF) –  $M \times N \times 4$ . Каждая ячейка такой матрицы будет иметь величину в зависимости от типа изображения. Например, для 8-битного серого, для которого каждая точка кодируется одним байтом, или 24-битного RGB цветного изображения, каждая точка которого кодируется тремя байтами, значения соответствующих чисел будут находиться в диапазоне изменения от 0 до 255. Для 16-битного серого или 48-битного цветного изображений в каждой ячейке, содержащей информацию о яркости белого или яркости одной из трех компонент цветного изображения, будут находиться 2 байта. Если в переменной находится черно-белое изображение, то каждая ячейка будет содержать 1 бит информации. Количество бит информации, записываемое в каждую ячейку, будет также зависеть от соответствующих форматов. Более подробные сведения об этой функции могут быть получены вызовом соответствующей справки командой **doc imread**.

Указанная функция поддерживает графические форматы BMP, GIF, ICO, JPEG, PCX, PNG, TIFF и другие.

После чтения изображения его следует просмотреть для контроля. Для этого имеются две функции: **imshow** и **imtool**. Первая выводит окно с изображением, в котором возможно провести с последним некоторые базовые действия, такие, как, например, файловые операции, копирование, вставка, поворот, вывод характеристик пикселей и др. Вторая, называемая **Image Tool**, позволяет осуществлять те же, что и **imshow**, но также и некоторые операции по обработке изображений, такие, как выделение и вырезку участков изображения, получение информации о них, определение расстояний между двумя точками и др. Применим **imshow** и получим изображение, показанное на рис. 4.2:

■ `imshow(I)`

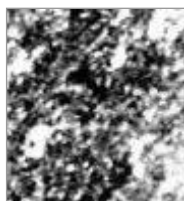


Рис. 4.2. Изображение аншлифа железистого кварцита

Для дальнейших действий преобразуем это цветное изображение в серое с помощью функции **rgb2gray**. Двойка здесь созвучна английскому предлогу **to**, означающему «в».

```
L=rgb2gray(I);  
figure(1),imshow(L);
```

Вторая строка выводит изображение в новое окно. При черно-белой печати его вид аналогичен представленному на рис. 4.2.

Существуют и другие функции преобразования изображений, как, например, **ind2gray**, **ntsc2rgb**, **rgb2ind**, **rgb2ntsc**, **mat2gray**, которые здесь специально рассматриваться не будут. Информация по ним может быть получена с помощью справки.

### Получение информации об изображениях

Можно проверить информацию о переменных **I** и **L**, в которых содержатся изображения. Она отображается в окне рабочего пространства **MATLAB**. Из нее следует, что структура переменной **I**, соответствующей цветному изображению, имеет вид  $97 \times 89 \times 3$  ячеек формата **unit8**, т. е. в каждой из них содержится по одному байту. Значения чисел лежат в диапазоне от 0 до 255 единиц. В отличие от нее переменная **L** имеет структуру  $97 \times 89$  аналогичных ячеек. Такое отличие структур обусловлено тем, что первая переменная содержит цветное изображение, а вторая - серое. Аналогичная информация может быть получена также командой **whos**.

### Построение гистограммы яркости и регулировка контрастности

Важнейшей характеристикой изображения является его контрастность. Она может быть оценена по соотношению границ распределения яркостей пикселей изображения и полным диапазоном, в котором они могут быть распределены. Последний определяется типом переменных. Например, для переменных формата **unit8** полный диапазон находится в пределах от 0 до 255 единиц. Для оценки контрастности может быть использована гистограмма яркостей, которая строится с помощью функции **imhist** следующим образом:

```
figure  
imhist(L)
```

Как и раньше, первая строка служит для вывода графика в отдельном окне, они могут быть записаны и в одной строке через запятую. Вторая строка вызывает вывод графика, показанного на рис. 4.3.

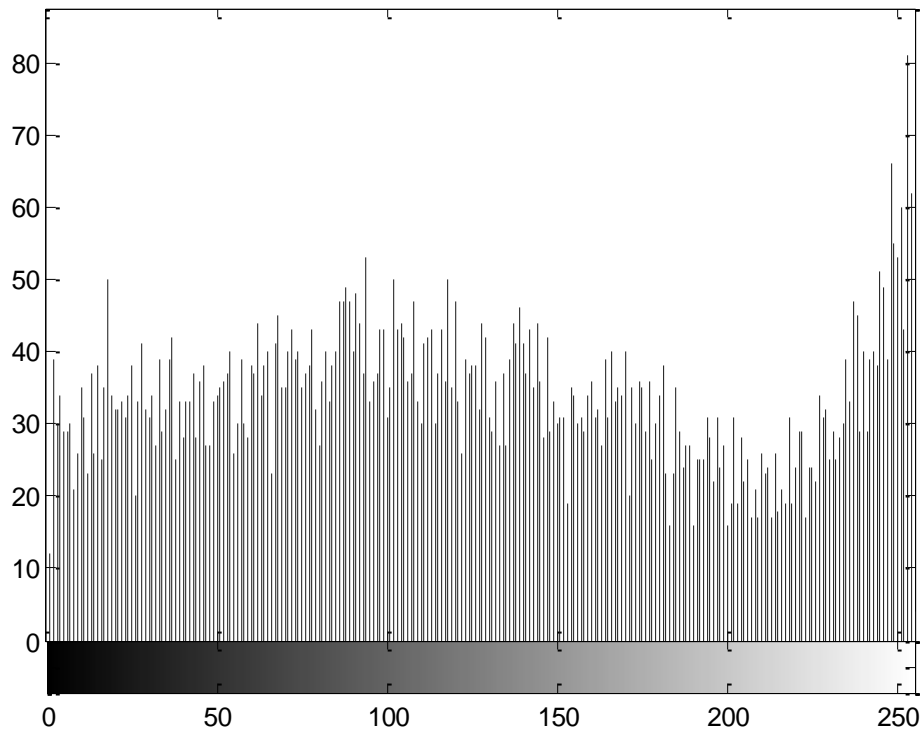


Рис. 4.3. Гистограмма яркостей изображения, выводимая функцией **imhist**

По оси абсцисс здесь откладывается яркость пикселей. При этом 0 соответствует самому черному, а 255 – самому белому изображению. По оси ординат откладывается количество пикселей с соответствующей яркостью.

Как следует из гистограммы рис. 4.3, яркости в изображении распределены по всему диапазону [0, 255]. Это говорит о том, что весь диапазон яркостей используется в достаточной степени, и изображение обладает хорошей контрастностью. Если бы распределение яркостей изображения находилось в более узких границах, то такое изображение было бы малоконтрастным. Улучшить контрастность можно с помощью функции **histeq**, название которой образовано от сочетания *histogram equalization*, т. е. выравнивание контраста, а соответствующая строка программного кода имеет вид

```
L2 = histeq(L);  
figure, imshow(L2)
```

Во второй строке, как и раньше, результат выравнивания выводится в новом окне.

Для улучшения контраста имеются также функции **imcontrast**, **imadjust**, **adapthisteq** и другие, справку по которым можно получить обычным образом. Здесь они отдельно рассматриваться не будут.

### Запись преобразованного изображения на диск

Запись изображения в одном из графических форматов осуществляется функцией **imwrite** следующим образом

```
imwrite (L2, 'magpart.png');
```

В данном случае изображение записывается в формате **Portable Network Graphics (PNG)**, но могут быть использованы и другие форматы.

### Проверка характеристик созданного файла

Получить данные по изображению, записанному в файл, можно с помощью вызова функции **imfinfo** следующим образом:

```
imfinfo('magpart.png')
```

При этом выводятся данные по формату, размеру и др. Например, файл **magpart.png** имеет следующие характеристики

```
ans =  
      Filename: 'magpart.png'  
      FileModDate: '03-янв-2010 12:38:51'  
      FileSize: 7762  
      Format: 'png'  
      FormatVersion: []  
      Width: 89  
      height: 97  
      BitDepth: 8  
      ColorType: 'grayscale'  
      FormatSignature: [137 80 78 71 13 10 26 10]  
      Colormap: []  
      Histogram: []  
      InterlaceType: 'none'  
      Transparency: 'none'  
      SimpleTransparencyData: []  
      BackgroundColor: []  
      RenderingIntent: []  
      Chromaticities: []  
      Gamma: []  
      XResolution: []
```

```
YResolution: []
ResolutionUnit: []
XOffset: []
YOffset: []
OffsetUnit: []
SignificantBits: []
ImageModTime: '3 Jan 2011 09:38:51 +0000'
Title: []
Author: []
Description: []
Copyright: []
CreationTime: []
Software: []
Disclaimer: []
Warning: []
Source: []
Comment: []
OtherText: []
```

Как видно, данная функция позволяет вывести большое количество разнообразных данных, из которых для файла **magpart.png** указана только небольшая часть.

В заключение приведем полный текст рассмотренной программы.

```
close all;
clear;
img='magpart.jpg';
I=imread(img);
imshow(I);
L=rgb2gray(I);
figure(1),imshow(L);
figure
imhist(L)
L2 = histeq(L);
figure, imshow(L2)
imwrite (L2, 'magpart.png');
imfinfo('magpart.png')
```

#### 4.4. Определение содержания минералов в горной породе

Рассмотрим одну из задач, часто встречающуюся в горном деле. Анализ минерального состава горных пород является одной из основных операций, позволяющих получить характеристики полезных ископаемых. Для этой цели используются физические и химические методы. Покажем, как эта задача может быть решена с помощью анализа изображений аншлифов горных пород,

которые могут быть получены в электронной форме путем сканирования на обычных или специальных сканерах.

Для решения указанной задачи будет использовано цветное изображение аншлифа железистого кварцита. Предварительно минеральный состав известен, при этом отражательные способности каждого компонента различны, что будет использовано для распознавания соответствующих областей на изображении. Основными компонентами здесь являются магнетит, имеющий диапазон яркостей от 0 до 60 единиц, кварц с яркостью от 140 до 256 единиц, а также гематит и другие минералы, занимающие промежуточный диапазон от 61 до 139 единиц. Установление этих границ осуществляется из предварительных измерений, анализа гистограмм яркостей, а также сравнения результатов анализа изображений с результатами, полученными с помощью физических или химических методов. В данном случае будет проведено распознавание и определение содержания минералов в трех областях, содержащих магнетит, кварц и другие минералы. При выполнении программы будет определено количество пикселей, общее для всего изображения, количество пикселей, соответствующих каждой из упомянутых областей, а затем содержание каждого минерала будет определено как отношение соответствующего количества пикселей к общему их числу.

Начинается программа с команд очистки различных областей системы **MATCAD**. Напомним, что при вводе данных строк программного кода комментарии, следующие за знаком процента %, и сам этот знак можно не вводить.

```
clc          % Очистка экрана
clear all    % Очистка рабочей области памяти
close all    % Заккрытие окон с изображениями
```

Далее следует прочесть данные из файла и вывести изображение для просмотра.

```
% Чтение изображения из файла в матрицу IM
P='magpart.jpg'
IM = imread(P);
figure, imshow(IM), title('Исходное изображение');
```

Распознавание областей ведется по яркости серого изображения. Поскольку исходное изображение в матрице **IM** цветное, его следует преобразовать в серое **DI**, после чего осуществить вывод на экран для контроля.

```
% Преобразование цветного изображения в серое
DI = rgb2gray(IM);
figure, imshow(DI), title('Серое изображение');
```

Результат действия этого участка программного кода показан на рис. 4.4.



Рис. 4.4. Результат преобразования цветного изображения в серое

Для выделения определенных областей изображения следует задать границы яркости каждого минерала.

```
% Границы яркости магнетита и кварца  
magmin=0;      % минимальная яркость магнетита  
magmax=60;     % максимальная яркость магнетита  
quartzmin=140; % минимальная яркость кварца  
quartzmax=256; % % максимальная яркость кварца
```

Здесь заданы границы яркости магнетита и кварца, границы других минералов находятся между ними.

Далее нужно выделить области изображения, соответствующие каждому минералу. В результате этой операции будут получены три бинарные матрицы, содержащие данные об искомым областях. Они будут содержать 1 на участках, представляющих интерес, где яркость укладывается в заданный диапазон, и 0 на других. Эта операция будет осуществлена с помощью функции **roicolor**. Данная функция позволяет выделить в индексированной (т. е. содержащей 0 и 1) или яркостной (серой или цветной) матрице область, значения чисел в ячейках которой находятся в заданном диапазоне. Входные переменные в **a** - численные, а выходные – логические. Эта функция имеет следующий формат:

```
BW = roicolor(A,low,high)
```

Здесь **bw** – выходная бинарная матрица, **A** – входная бинарная или числовая матрица, **low**, **high** – границы яркости областей, которые следует выделить.

Для нашего случая программный код будет иметь вид

```
% содержание магнетита  
BW = roicolor(DI,magmin,magmax);  
figure, imshow(BW), title('Участки черного');
```

Результат показан на рис. 4.5, а.



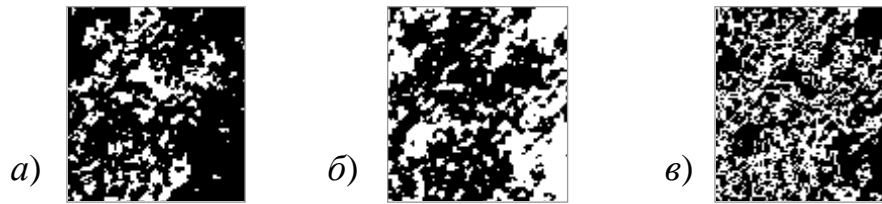


Рис. 4.5. Белым цветом отмечены области, содержащие агрегаты магнетита (а), кварца (б), других минералов (в)

Далее следует подсчитать количество пикселей этого изображения, содержащих 1.

```
% Расчет площадей
D=size(BW);
S=0;
for i=1:D(1),
    for j=1:D(2),
        if BW(i,j)==1
            S=S+1;
        end
    end
end
disp('Магнетит')
S;
```

В первой строке здесь определяется размер матрицы, при этом **D** представляет собой вектор, первый элемент которого **D(1)** показывает количество строк, а второй элемент **D(2)** – количество столбцов в матрице изображения. Во второй строке организуется счетчик **S**, которому присваивается начальное значение, равное нулю. Далее в двойном цикле, перебирающем все пиксели изображения, в этот счетчик прибавляются единицы, если в данной ячейке находится единица, что достигается с помощью логической операции **if BW(i,j) == 1**. Двойной знак «равно» здесь означает логическое сравнение.

Оператор **disp('Магнетит')** в предпоследней строке выводит в командное окно MATLAB надпись **Магнетит**, а последняя строка – количество пикселей, соответствующих области магнетита.

Для подсчета относительного содержания магнетита необходимо знать общее количество пикселей в изображении. Для этого нужно просто перемножить количество строк и столбцов матрицы **D**. Деление количества пикселей **S**, соответствующих магнетиту, на общее их число **SM**, дает относительное содержание магнетита. В данном случае эта величина выражена в процентах.

```
disp('Всего')
SM=D(1)*D(2);
```

```
disp('Содержание магнетита, %')
S/SM*100
```

Аналогичные действия проводятся для кварца и других минералов.

```
%Содержание кварца
BW = roicolor(DI,quartzmin,quartzmax);
figure, imshow(BW), title('Кварц');

S=0;
for i=1:D(1),
    for j=1:D(2),
        if BW(i,j) == 1
            S=S+1;
        end
    end
end
disp('Кварц')
S;
disp('Содержание кварца, %')
S/SM*100

%Содержание других минералов
BW = roicolor(DI,magmax+1,quartzmin-1);
figure, imshow(BW), title('Участки других минералов (гематит и др.)');

S=0;
for i=1:D(1),
    for j=1:D(2),
        if BW(i,j) == 1
            S=S+1;
        end
    end
end
disp('Другие минералы (гематит и др.)')
S;
disp('Содержание других минералов, %')
S/SM*100
```

Приведем текст программы, собранный воедино.

```
P='magpart.jpg'
IM = imread(P);
figure, imshow(IM), title('Исходн. изображение');
DI = rgb2gray(IM);
```

```

figure, imshow(DI), title('Серое изображение');
magmin=0;
magmax=60;
quartzmin=140;
quartzmax=256;
BW = roicolor(DI,magmin,magmax);
figure, imshow(BW), title('Участки черного');
D=size(BW)
S=0;
for i=1:D(1),
    for j=1:D(2),
        if BW(i,j)==1
            S=S+1;
        end
    end
end
disp('Магнетит')
S;
disp('Всего')
SM=D(1)*D(2);
disp('Содержание магнетита, %')
S/SM*100
BW = roicolor(DI,quartzmin,quartzmax);
figure, imshow(BW), title('Участки белого');
S=0;
for i=1:D(1),
    for j=1:D(2),
        if BW(i,j)==1
            S=S+1;
        end
    end
end
disp('Кварц')
S;
disp('Содержание кварца, %')
S/SM*100
BW = roicolor(DI,magmax+1,quartzmin-1);
figure, imshow(BW), title('Участки других (гематит и др.)');
S=0;
for i=1:D(1),
    for j=1:D(2),
        if BW(i,j)==1
            S=S+1;
        end
    end
end

```

```
end
disp('Другие минералы (гематит и др.)')
S;
disp('Содержание других минералов, %')
S/SM*100
```

Данное исследование имеет своей целью определение содержания минералов в относительных единицах, поэтому здесь не учитываются размеры измеряемых областей.

#### 4.5. Определение размеров минеральных агрегатов

Имеется ряд задач, в которых необходимо рассматривать размеры областей. В этом случае необходимо учитывать разрешающую способность, с которой было получено исходное изображение. К таким задачам относится определение распределения размеров или площади минеральных агрегатов.

Дальнейшее освоение приемов обработки изображений будут рассмотрены на примере определения площадей агрегатов магнетита по изображению аншлифа железистого кварцита, показанного на рис. 4.2.

Программа начинается с уже знакомых команд очистки

```
clc % Очистка экрана
clear all % Очистка рабочей области памяти
close all % Заккрытие окон с изображениями
```

Чтение исходного изображения и его вывод для контроля

```
fil='magpart.jpg';
I=imread(fil);
```

Цветное изображение преобразуется в серое, размещается в матрице L и его вид выводится на экран

```
L=rgb2gray(I);
figure, imshow(L);
```

Внешний вид серого изображения показан на рис. 4.4.

В данном случае изображение имеет хороший контраст. В случаях мало-контрастных изображений требуется корректировка их контрастности, которая может быть осуществлена с помощью функции **imadjust**.

```
L = imadjust(L);
```

## Определение размера матрицы изображения

```
|| [N M]=size(L);
```

Создание матрицы того же размера для последующего заполнения и образования бинарной матрицы **Lbw**

```
|| Lbw=L;
```

Бинаризация изображения по порогу **thr**, который определяется из анализа гистограммы. В данном случае в отличие предыдущего параграфа, где для этой цели применялась функция **roicolor**, здесь использован другой вариант бинаризации, более открытый для понимания, как происходит преобразование изображения

```
|| thr = 30;  
|| for i=1:N;  
||     for j=1:M;  
||         if L(i,j)>thr;  
||             Lbw(i,j)=0;  
||         else  
||             Lbw(i,j)=1;  
||         end;  
||     end;  
|| end;  
|| figure, imshow(Lbw,[0 1]);
```

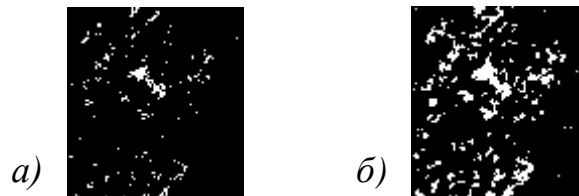


Рис. 4.6. Изображения бинарной матрицы при пороге  $thr=10$  (а) и  $thr=30$  (б)

На выходе этой операции образуется матрица **Lbw**, в ячейках которой только единицы и нули. При этом если в ячейке число, характеризующее яркость, больше порога **thr**, то в бинарную матрицу записывается 0, т. е. она не относится к магнетиту. Если же это число меньше порога **thr**, то записывается 1, т. е. считается, что изображение в этой ячейке соответствует магнетиту. Результаты такой операции при порогах 10 единиц и 30 единиц представлены на рис. 4.6. При более высокой верхней границе диапазона яркостей, к которому относятся элементы изображения, количество последних будет больше, что приводит к большему размеру площадей светлых участков на рис. 4.6, б, по сравнению с изображением на рис. 4.6, а. Из представленных изображений

следует, что выбор правильного порога при переводе в бинарную матрицу является очень важной задачей.

Для определения параметров структурных элементов, которые в нашем случае в матрице **Lbw** отмечены единицами, в версиях, начиная с 2009 г., используется функция **bwconncomp**. Эта функция осуществляет поиск компонентов в бинарном изображении любой размерности. Она имеет формат

$$cc = bwconncomp(bw),$$

где **cc** – выходная матрица, имеющая структуру **cc**, название которой происходит от слов «**connected components**» (соединенные компоненты), **bw** – входная бинарная матрица с анализируемым изображением. Структура **cc** имеет четыре поля:

Поле	Описание
Connectivity	Связность соединенных компонентов (объектов)
ImageSize	Размер входной матрицы
NumObjects	Количество связанных компонентов (объектов) во входной матрице
PixelIdxList	Массив ячеек размера $1 \times \text{NumObjects}$ , где $k$ -тый элемент в массиве есть вектор, содержащий линейные индексы (номера) пикселей в $k$ -том объекте

По умолчанию функция **bwconncomp** имеет 8 связанных компонентов для двумерной входной матрицы и 26 для трехмерной. Существует также вариант для многомерной входной матрицы, который здесь не рассматривается. В расширенном варианте эта функция имеет формат

$$cc = bwconncomp(bw,conn)$$

где помимо описанных выше составляющих **conn** – скалярная величина, задающая количество связей при поиске объектов. Этот параметр может принимать следующие значения в зависимости от размерности входной матрицы, приведенные в табл. 3.1.

Таблица 3.1.  
Значения параметра **conn** и их описание

Значение параметра	Описание
4	двумерное четырех-связанное соседство
8	двумерное восьми-связанное соседство
6	трехмерное шести-связанное соседство
18	трехмерное 18-связанное соседство
26	трехмерное 26-связанное соседство

Более подробно связность и соседство будут обсуждаться ниже. Матрица **bw** может иметь логический или численный формат и должна быть вещественной, двумерной 2-D и неразрезанной, формат структуры **cc** описан выше. Для нашего случая программный код с указанной функцией имеет вид

```
|| cc = bwconncomp(Lbw);
```

Результатом действия этого кода будет создание матрицы **cc**. Ее содержимое можно вывести в командное окно, записав (необязательная строка)

```
|| labeled = labelmatrix(cc)
```

**Примечание.** При использовании предыдущих версий системы MATLAB (2006 и ранних) функция **bwconncomp**, отсутствующая в этих версиях, может быть заменена функцией **bwlabel**, имеющей формат

$$L = \text{bwlabel}(\text{bw}, n)$$

где **L** – выходная матрица, **bw** – входная матрица черно-белого изображения, **n** – количество элементов, по которым будет определяться связность, может принимать значения 4 или 8, по умолчанию 4. Нули обозначают фон, а все другие величины обозначают пронумерованные объекты интереса.

Возможен также следующий формат записи (**рекомендуется при использовании предыдущих версий системы Mathcad**):

```
|| [L,num] = bwlabel(bw,n);
```

где **L** – выходная матрица, **num** – количество пронумерованных элементов интереса, **n** – общее количество элементов. Пример части содержимого матрицы **cc** приведен на рис. 4.7.

0	0	0	0	0	0	0	0	25	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	26	26	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	20	0	0	27	0	0	0
0	0	0	0	0	0	20	20	0	0	0	0	0
0	0	0	0	0	20	20	20	0	0	0	0	0
0	0	0	0	0	0	0	20	0	0	0	0	0
0	16	0	0	0	0	0	0	0	0	0	0	0
0	0	16	16	0	0	0	0	0	0	0	0	0
0	16	0	16	0	0	0	0	0	0	0	0	0
0	16	0	0	0	21	21	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	17	0	17	0	0	0	0	0	0	0	0	0
0	0	17	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	24	0	0	0	0	0

Рис. 4.7. Пример части содержимого матрицы, в ячейки которой записаны номера обнаруженных объектов, эти ячейки отмечены рамками

Для расчетов параметров областей вывод такой матрицы необязателен и приведен здесь для понимания алгоритма обработки изображений. Чтобы получить значения набора параметров найденных областей, используется функция **regionprops**. Ее формат

```
stats = regionprops(cc,properties)
```

где **stats** – статистические параметры, задаваемые переменной **properties**, **cc** – переменная, содержащая изображение. В качестве последней могут использоваться:

- бинарная логическая матрица **BW**;
- структура **CC**, образуемая на выходе функции **bwconncomp** пример которой приведен на рис. 4.7;
- матрица **L** с маркированными областями объектов, схожая со структурой **CC**.

Параметр **properties** (свойства) может быть списком строковых переменных, разделенных запятой, массив (матрица) ячеек, содержащих строковые переменные, строку **'all'**, или строку **'basic'**. Набор строковых переменных, означающих измеряемые величины, включает в себя параметры формы и параметры пиксельных величин (для серых изображений).

Здесь упомянем только параметры формы:

'Area'	'EulerNumber'	'Orientation'
'BoundingBox'	'Extent'	'Perimeter'
'Centroid'	'Extrema'	'PixelIdxList'
'ConvexArea'	'FilledArea'	'PixelList'
'ConvexHull'	'FilledImage'	'Solidity'
'ConvexImage'	'Image'	'SubarrayIdx'
'Eccentricity'	'MajorAxisLength'	
'EquivDiameter'	'MinorAxisLength'	

В рассматриваемом примере интерес будет представлять параметр **'Area'**, означающий площадь. Если в качестве параметра **properties** указано **'all'**, функция **regionprops** дает на выходе все измеренные параметры формы. Если свойства не указаны или в качестве этого параметра указано **'basic'**, то рассчитываются величины **'Area'**, **'Centroid'** (центроид, т. е. центр тяжести фигуры), и **'BoundingBox'** (ограничивающая рамка).

Итак, следующая строка программного кода, в которой осуществляется расчет площадей обнаруженных объектов, имеет вид

```
█ graindata = regionprops(cc, 'basic');
```



Теперь следует выделить из всего набора параметров только площади найденных объектов. Это делается с помощью строки

```
grain_areas = [graindata.Area];
```

Переменная **grain\_areas** содержит данные о площадях объектов, в ней в ячейке с определенным номером содержится число, равное площади объекта в пикселях. Теперь становится возможным построить гистограмму распределения площадей объектов.

```
nbins = 20;  
figure, hist(grain_areas, nbins)  
title('Распределение площадей агрегатов магнетита');
```

В первой строке присваивается численное значение переменной `nbins`, означающей количество разрядов гистограммы. Во второй строке осуществляется ее построение в отдельном окне, а в третьей выводится заголовок. Внешний вид окна с выведенной гистограммой представлен на рис. 4.8.

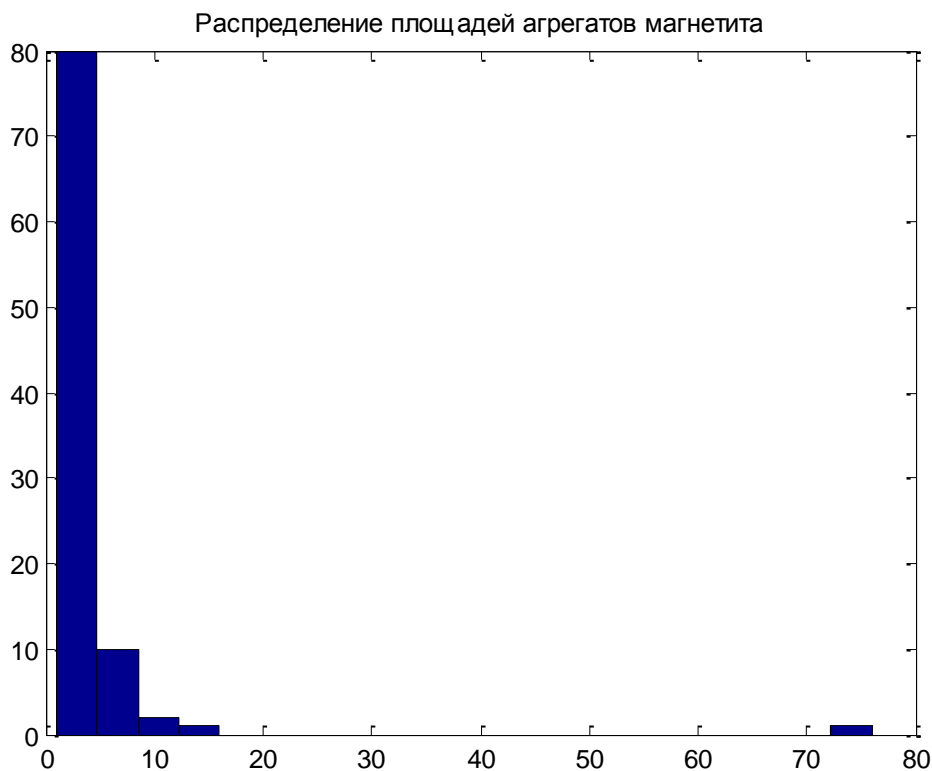


Рис. 4.8. Гистограмма распределения площадей (в пикселях) минеральных агрегатов магнетита

Здесь речь идет о площадях, измеренных в квадратных пикселях. Для получения эквивалентного размера объекта (например, в виде квадрата) в общепринятых единицах длины следует из площади извлечь квадратный корень и затем разделить полученный результат на разрешающую способность. Это достигается с помощью следующих строк программного кода.

```

graindata = regionprops(cc, 'basic');
grain_lin = sqrt([graindata.Area])/1600*25.4;
nbins = 20;
figure, hist(grain_lin, nbins)
title('Распределение размеров агрегатов магнетита');
xlabel('Размер агрегата (зерна), мм')
ylabel('Количество зерен в разряде')

```

Результаты расчетов и построений для порогового уровня **thr=60** приведены на рис. 4.9. При этом учтено, что изображение отсканировано с разрешающей способностью 1600 точек на дюйм. Это соответствует расстоянию между пикселями 0,0159 мм, при этом окончательный результат получен в метрических единицах измерения, в данном случае в миллиметрах.

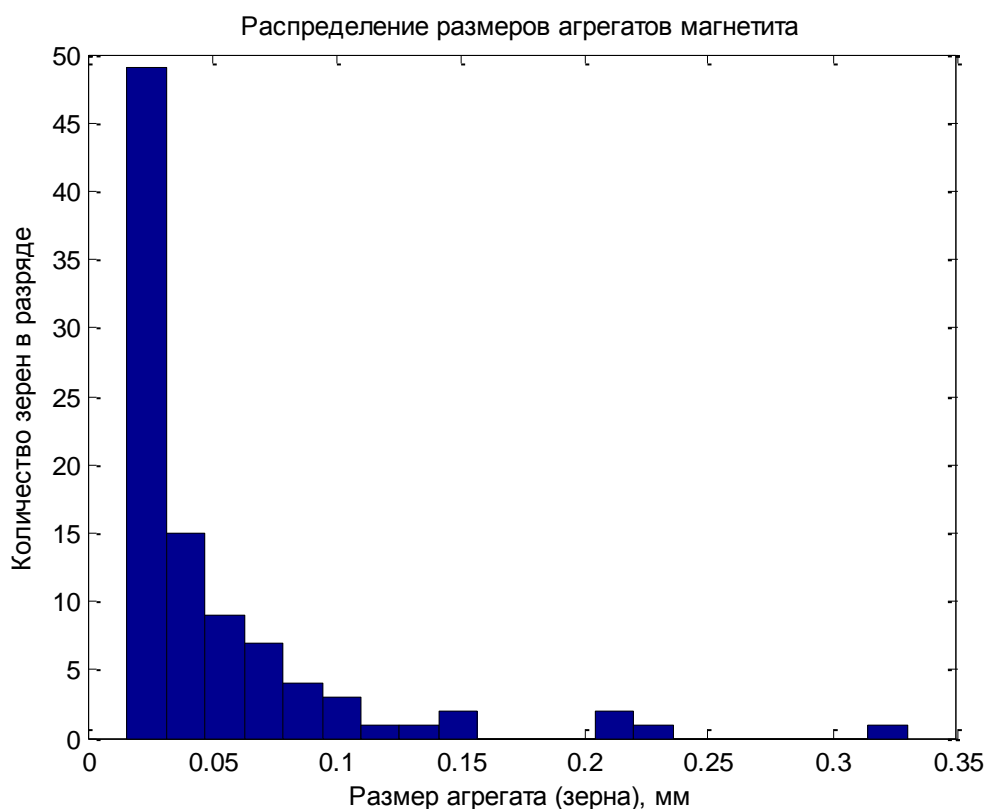


Рис. 4.9. Гистограмма распределения эквивалентных размеров зерен магнетита

Из этой гистограммы следует, что частицы магнетита имеют, в основном, размер менее 0,1 мм. Отдельные области демонстрируют большие размеры.

Важную роль в научных исследованиях играет так называемая валидация модели, т. е. проверка соответствия ее реальности. Проанализируем результаты, сравнив их с данными, которые получены традиционными методами.

Как отмечается в ряде публикаций по исследованию железистых кварцитов, в частности, в работах Н. М. Чернышова и Т. П. Коробкиной, зерна магнетита чаще всего имеют размер от 0,01 до 0,1 мм; агрегаты их достигают размера 0,15–0,3 мм. Распределение размеров, находящихся в диапазоне от 0,02 мм

до 0,1 мм, дает основание предположить, что в данном случае речь идет о зернах магнетита. На рис. 4.9 объекты размером свыше 0,1 мм могут быть интерпретированы как агрегаты, состоящие из нескольких зерен.

Текст программы определения размеров минеральных зерен и их агрегатов в миллиметрах, собранный воедино, приведен ниже.

```
clc % Очистка экрана
clear all % Очистка рабочей области памяти
close all % Закрытие окон с изображениями
fil='magpart.jpg';
I=imread(fil);
L=rgb2gray(I);
figure, imshow(L);
L = imadjust(L);
[N M]=size(L);
Lbw=L;
thr = 30;
for i=1:N;
    for j=1:M;
        if L(i,j)>thr;
            Lbw(i,j)=0;
        else
            Lbw(i,j)=1;
        end;
    end;
end;
figure, imshow(Lbw,[0 1]);
cc = bwconncomp(Lbw); % при использовании версии 2009 и поздних
labeled = labelmatrix(cc) % при использовании версии 2009 и поздних
[cc,num] = bwlabel(bw,n) % при использовании версии 2006 и ранних
graindata = regionprops(cc, 'basic');
grain_lin = sqrt([graindata.Area])/1600*25.4;
nbins = 20;
figure, hist(grain_lin, nbins)
title('Распределение размеров агрегатов магнетита');
xlabel('Размер агрегата (зерна), мм')
ylabel('Количество зерен в разряде')
```

## КОНТРОЛЬНЫЕ ВОПРОСЫ К ГЛАВЕ 4

1. Какие аппаратные устройства используются для ввода изображений в компьютер?
2. Через какой интерфейс как правило подключаются устройства ввода изображений в компьютер?
3. Какие характеристики сканера и каким образом должны быть учтены при выборе сканера для ввода аншлифов горных пород или других изображений?
4. Что такое прямая и интерполяционная разрешающая способности при сканировании изображений?
5. Какие операции предусматривает обработка изображений?
6. Как осуществляется чтение и запись изображений при работе с файлами данных?
7. Перечислите основные программные функции обработки изображений, какие параметры необходимы для их работы?
8. Каким образом осуществляется построение гистограммы изображения и производится регулировка контрастности при ее недостаточной степени?
9. С помощью какой функции можно получить информацию об изображении?
10. Какая программная функция осуществляет перевод цветного изображения в серое?
11. Каким образом осуществляется выделение участков изображения с заданным диапазоном яркости?
12. Что такое связность соединенных компонентов и каким образом осуществляется выделение областей, принадлежащих одной группе?

## 5. АВТОМАТИЗИРОВАННЫЙ СБОР ДАННЫХ В ФИЗИЧЕСКОМ ЭКСПЕРИМЕНТЕ

### 5.1. Принципы компьютерного сбора данных в физическом эксперименте

Компьютер как универсальное средство обработки информации позволяет записывать в цифровой форме аналоговые сигналы, получаемые в процессе физического эксперимента. На рис. 5.1 схематически показан путь прохождения сигнала при его вводе в компьютер.

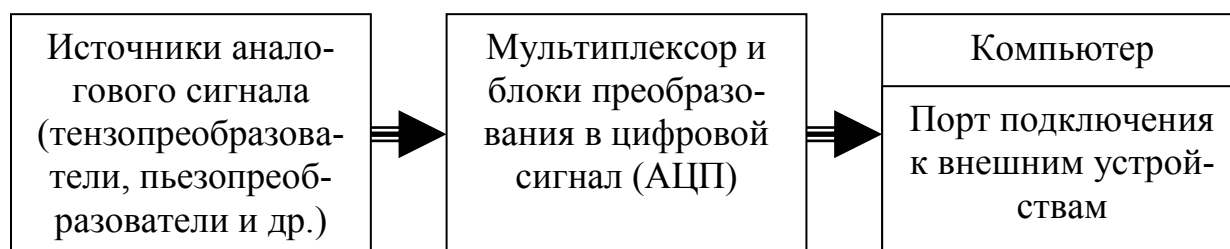


Рис. 5.1. Схема прохождения сигналов преобразователей физических величин при их вводе в компьютер

Источниками сигнала как правило являются аналоговые датчики, т. е. такие, на выходе которых образуется сигнал в виде электрических напряжений или токов, пропорциональных измеряемой величине. В последнее время аналоговые датчики стали совмещаться с аналого-цифровыми преобразователями, на выходе которых уже получают двоичный последовательный код. Такие датчики можно подключать через линию связи либо к цифровому контроллеру, либо непосредственно к стандартному цифровому входу компьютера.

Если в качестве источников сигналов используются аналоговые датчики, то они подключаются к входу аналого-цифрового преобразователя, цифровой выход которого различными способами может быть соединен с компьютером. Если используется несколько датчиков, то они:

- либо подключаются каждый к своему АЦП, и ввод в компьютер сигналов от различных датчиков осуществляется в цифровой форме, при этом выходы АЦП опрашиваются последовательно внутри компьютера;
- либо к одному АЦП через аналоговый мультиплексор – устройство с несколькими входами и одним выходом; в этом случае мультиплексор последовательно во времени подключает выход каждого датчика к входу АЦП, и данные от датчиков на АЦП передаются друг за другом, а в компьютер они поступают с его выхода последовательно.

При программировании таких устройств ввода аналоговой информации указывается либо номер конкретного входа, либо включается режим циклического опроса входных каналов.

При преобразовании аналоговых сигналов должно быть учтено соотношение

$$f_{\partial} \geq 2 \cdot f_{\max},$$

где  $f_d$ ,  $f_{\max}$  – частота дискретизации и максимальная частота спектра. Это теоретическое соотношение, в действительности частоту дискретизации выбирают еще выше. Например, при преобразовании сейсмоакустических сигналов, спектр которых находится в диапазоне от нескольких герц до 5 килогерц, частота дискретизации может быть выбрана из условия  $f_d \geq 2 \cdot 5 = 10$  кГц, к примеру, 22 кГц или 44,1 кГц, находящихся среди стандартных значений, которые могут быть установлены в звуковой карте компьютера.

В зависимости от количества каналов измерения, скорости изменения входных сигналов или ширины их спектра, требуемой детальности или динамического диапазона аппаратная часть, размещаемая между источниками этих сигналов и компьютером, может быть различна по сложности. В самом простейшем случае сигналы от внешних преобразователей, например, от датчиков вибраций, могут быть введены через звуковую карту компьютера, которая в своем составе содержит АЦП. Она имеет два канала, через них к компьютеру возможно подключить один (в монорежиме) или два (в стереорежиме) преобразователя. В последнем случае регистрация ведется по обоим каналам параллельно. Частота дискретизации задается программным образом.

Функционально система сбора данных с датчиков на компьютере напоминает осциллограф. У последнего возможны два режима регистрации сигналов - непрерывный и ждущий с запуском по превышению порога срабатывания. Однако компьютерная информационно-измерительная система имеет некоторые особенности, и для реализации режимов осциллографа необходимо написать программу. Как это сделать, будет рассмотрено позднее.

В режиме ждущего запуска есть возможность регистрации с предысторией, когда записываются не только данные, пришедшие после превышения порога, но и предшествующие отсчеты, количество которых задается заранее.

При подготовке эксперимента, в котором нужно производить регистрацию сигналов каких-либо преобразователей, следует задать ряд параметров, определяющих режим работы регистрирующего устройства. При регистрации сигналов от акселерометров или аналогичных датчиков такими параметрами являются:

- количество входных каналов и их номера, по которым будет производиться регистрация;
- частота дискретизации, ее выбор описан выше;
- диапазон входных сигналов или усиление;
- диапазон выходных величин АЦП, которые будут записаны.

Кроме того, при ждущем режиме работы задают:

- уровень порога срабатывания начала записи при ждущем режиме регистрации;
- количество отсчетов «предыстории» сигнала, т. е. участка записи до момента срабатывания по превышению порога;
- количество отсчетов сигнала, которое будет записано после срабатывания по превышению порога.

Прежде чем разрабатывать и устанавливать измерительную систему сбора данных, следует понять, какие физические величины вы хотите изме-

рять, характеристики этих физических величин, какие преобразователи нужно применить для этого, а также какие измерительные устройства следует применять.

Каждый новый эксперимент сопровождается:

- системными установками, т. е. установкой аппаратной и программной частей;
- калибровкой;
- пробным включением.

### Установка аппаратной и программной частей измерительной системы

Прежде всего следует установить и настроить аппаратную часть и ее программное обеспечение. Установка аппаратной части заключается в размещении на материнской плате компьютера или во внешнем шасси модулей АЦП, ЦАП или цифрового ввода/вывода. Программная часть предусматривает копирование на диск и установку в системе драйверов аппаратных модулей и прикладного программного обеспечения.

**Драйверы** – это особые программы, обеспечивающие взаимодействие операционной системы с аппаратными модулями. Прикладное программное обеспечение в процессе своей работы использует драйверы для обращения к этим модулям. Каждое внешнее устройство имеет свой драйвер. Некоторые драйверы для стандартных устройств уже предусмотрены в системе, подключаются автоматически и не требуют отдельной дополнительной установки. Если же используются нестандартные устройства, к которым относятся АЦП, то необходима отдельная установка драйверов. Ко многим звуковым картам, представляющим собой сочетание АЦП на сигнальном входе и ЦАП на выходе, драйверы имеются в системе и их отдельно устанавливать не надо.

После того, как аппаратная часть и программное обеспечение установлены на компьютер, к нему можно подключать преобразователи неэлектрических величин в электрический сигнал, называемые также датчиками. Таким датчиком, например, является микрофон, преобразующий в электрический сигнал звуковое давление воздуха и подключаемый к звуковой карте. Компьютер с установленным в него программным обеспечением, картой АЦП, а также с подключенными датчиками представляет собой информационно-измерительную систему.

### Калибровка

После подключения датчиков проводится калибровка такой системы, называемая также тарировкой или эталонировкой. Калибровка заключается в подаче на вход такой системы определенных сигналов (например, вибраций с известными параметрами) и записи выходных показаний. Для многих устройств сбора данных такого класса калибровка легко может быть осуществлена с помощью программного обеспечения, поставляемого изготовителем вместе с АЦП. Такая запись используется для пересчета выходных пока-

заний системы на записи в значения входной величины. Например, если входной величиной датчика является температура  $T$ , а выходной – единицы АЦП  $y$ , то в процессе тарировки получают зависимость  $T(y)$ , позволяющую пересчитать записанные значения  $y$  в величины температуры  $T$ . Такая зависимость используется затем в программах обработки данных, зарегистрированных в процессе эксперимента.

В технической документации для преобразователей часто дается значение коэффициента преобразования измеряемой величины в величину электрического сигнала, называемого также **чувствительностью**. Предположим, для датчиков-акселерометров она составляет  $K_d=10 \text{ мВ/г} \cong 0,001 \text{ В/(м/с}^2\text{)}$ . В этом случае тарируется сам АЦП. Для этого либо тарировка ведется путем подачи электрического напряжения на вход АЦП с последующей регистрацией показаний, либо используют расчетный коэффициент преобразования. Он может быть получен из следующих соотношений. Допустим, мы измеряем колебательное ускорение и используем 16-разрядный АЦП с диапазоном измерения  $\pm 0,1 \text{ В}$ , т. е. полный диапазон составляет  $U_{max} = 0,2 \text{ В}$ . Количество уровней квантования  $N = 2^{16} = 65536$ , тогда коэффициент передачи напряжения АЦП соответствует  $K_{\text{АЦП}} = N / U_{max} = 327680 \text{ 1/В}$ . Коэффициент пересчета показаний АЦП в единицы ускорения

$$K = \frac{1}{K_d K_{\text{АЦП}}} = \frac{1}{0,001 \frac{\text{В}}{\left(\frac{\text{м}}{\text{с}^2}\right)} \cdot 327680 \frac{1}{\text{В}}} \cong 3,05 \cdot 10^{-5} \text{ м/с}^2.$$

Например, если амплитуда ускорения на записи имеет величину 2000 единиц АЦП, то это соответствует ускорению  $2000 \cdot 3,05 \cdot 10^{-5} = 0,061 \text{ м/с}^2$ .

### Пробный запуск

После того, как аппаратная часть установлена и откалибрована, можно осуществлять регистрацию показаний. Если есть уверенность, что характеристики регистрируемых сигналов на записи соответствуют действительности, то тогда остается разместить датчики на исследуемом объекте и производить измерения и регистрацию их результатов.

Однако в реальности измерения могут сопровождаться помехами, что потребует экранировки датчиков, либо нужно увеличить частоту опроса датчиков, либо установить на входе антиалиасинговый фильтр, устраняющий частоты выше удвоенной частоты дискретизации. Эти факторы являются помехами для проведения точных измерений. Для устранения этих помех придется производить пробные регистрации и подбирать различные режимы для получения неискаженных данных.



## 5.2. Использование звуковой карты для ввода и записи аналоговых сигналов в компьютере с использованием системы MATLAB

### 5.2.1. Предварительные сведения о звуковой карте

В качестве аппаратного модуля, содержащего аналого-цифровой преобразователь и служащего для ввода аналоговых данных в компьютер, будем рассматривать звуковую карту как наиболее распространенную и имеющуюся в каждом компьютере. Она имеет также и цифроаналоговый преобразователь, однако здесь мы основное внимание будем уделять именно вводу сигналов датчиков в компьютер. Именно такой режим наиболее часто используется в экспериментах. Аналоговый выход применяется там, где осуществляется активный эксперимент, т. е. при котором задаются различные режимы испытаний, меняющиеся по программе, либо определенные сигналы используются для воздействия на объект исследования. Такие сигналы берутся с аналогового выхода карты.

Упрощенная функциональная схема звуковой карты приведена на рис. 5.2. Обычно она содержит микрофонный и линейный входы, выход, внутри нее находятся блоки усиления, АЦП.

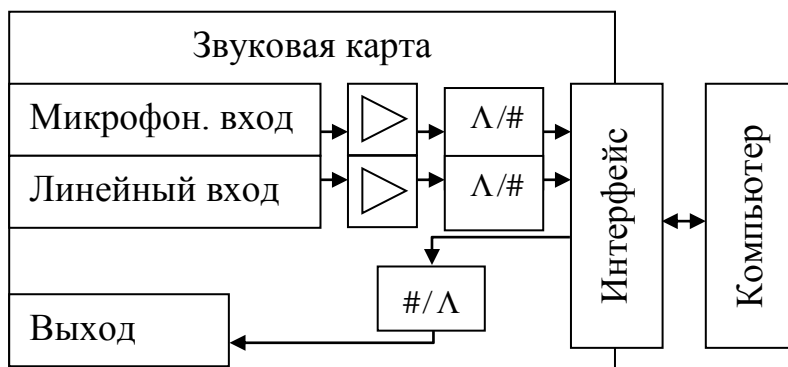


Рис. 5.2. Функциональная схема звуковой карты

Входы служат для подачи сигналов от микрофона или других источников сигнала. Микрофонный вход более чувствительный, чем линейный. Эти входы через программируемые усилители, которые обозначены треугольником, подключаются к АЦП, имеющим обозначение  $\Lambda/\#$ , а от них к компьютеру. Современные компьютеры

уже содержат звуковую карту в своем составе, но она может располагаться отдельно и подключаться, например, через порт **USB**. Это сокращение означает **universal serial bus**, что переводится как «универсальная последовательная шина».

При выводе сигналов выходные коды через цифроаналоговый преобразователь  $\#/ \Lambda$  в виде аналоговых сигналов подаются на выходной разъем звуковой карты.

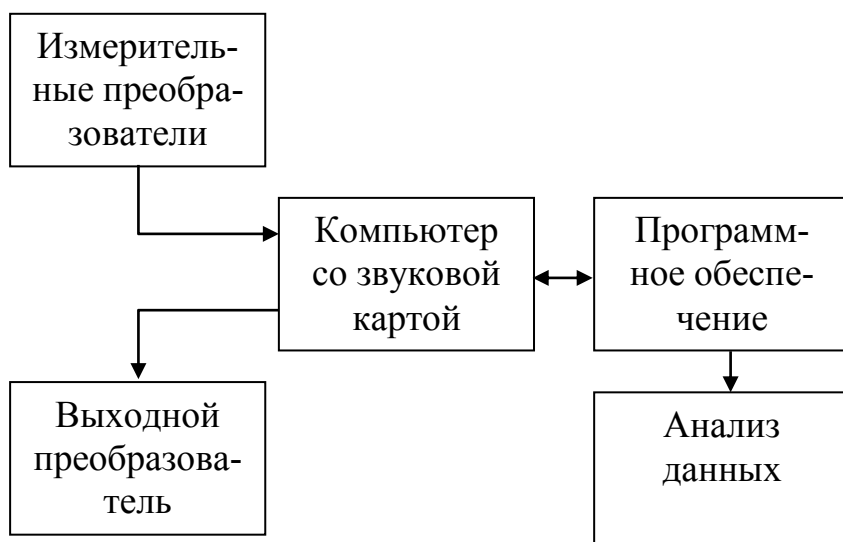


Рис. 5.3. Функциональная схема физического эксперимента

Общая функциональная схема аппаратно-программной части физического эксперимента представлена на рис. 5.3.

Сигналы от преобразователей, размещенных на исследуемом объекте, поступают на АЦП или звуковую карту, таким образом вводятся в компьютер и с помощью программного обеспечения, управляющего ре-

гистрацией, в процессе эксперимента записываются на диск в виде цифровых файлов. Эти данные анализируются либо непосредственно в ходе эксперимента, либо после его окончания. В некоторых активных экспериментах кроме пассивной регистрации данных осуществляется также и управление режимами или подача зондирующих сигналов с помощью выходных преобразователей.

Во многих научных экспериментах интерес в первую очередь представляет собой ввод сигналов в компьютер и их регистрация. Поэтому основное внимание мы будем уделять именно вводу сигналов через линейный или микрофонный входы.

### 5.2.2. Программирование модулей ввода/вывода

Ознакомление с процедурой ввода данных от датчиков в информационно-измерительных системах рассмотрим на примере системы MATLAB. Создание программ в системе MATLAB базируется на концепции так называемого **объектно-ориентированного программирования** (ООП). Эта концепция использует понятия класса и объекта. **Объект** в компьютерной программе - это то, что обладает определенным поведением и способом представления. **Класс** можно сравнить с чертежом, по которому создаются объекты. Говорят, что объект — это экземпляр класса. Объект размещается в адресном пространстве вычислительной системы и появляется при создании экземпляра класса, например, при создании и запуске программы на исполнение. Объект в нашем случае связан с определенным устройством, или, другими словами, аппаратным модулем, через который осуществляется ввод/вывод аналоговых или цифровых сигналов. В нашем примере таким аппаратным модулем является звуковая карта.

При написании программы в ее начале нужно предусмотреть создание объекта, а при окончании этой программы или ее фрагмента – его удаление.

Для работы с аппаратными блоками аналогового и цифрового ввода/вывода сигналов в системе MATLAB используется программный инструментарий сбора данных, называемый Data Acquisition Toolbox. Он содержит различные функции, позволяющие использовать все возможности звуковых карт и других аппаратных средств аналогового и цифрового ввода/вывода данных и сигналов. Данный курс носит ознакомительный характер. Поэтому, не затрагивая подробно все из имеющихся программных операторов, функций, команд, рассмотрим наиболее употребительные из них, позволяющие осуществлять регистрацию сигналов в эксперименте самым простым способом. В конце данного раздела приведен список программных функций, используемых для организации ввода/вывода сигналов. Это позволит получать более подробную информацию, вызвав соответствующую справку. Файл справки по этому разделу вызывается командой (знаки >> выводятся системой)

```
>>help daq
```

Здесь **daq** является сокращением от Data acquisition, что означает Сбор данных. При этом выводится полный список функций, которые могут быть использованы при программировании сбора данных с помощью аналого-цифрового преобразования, вывода данных с помощью цифроаналогового преобразования, а также ввода/вывода цифровых данных.

Для того, чтобы просмотреть код любой функции, нужно ввести команду

```
>>type function_name
```

Здесь **function\_name** – имя интересующей функции. Например,

```
>>type analoginput
```

Справка может быть вызвана командой

```
>>daqhelp function_name
```

Инструментарий Data Acquisition Toolbox содержит компоненты, указанные в табл. 5.1. Эти компоненты представлены на рис. 5.4 в составе программно-аппаратной структуры компьютерной измерительной системы.

В такой системе датчики подключены к аппаратному блоку, который через программный драйвер связан с остальной программной частью системы. Если используется не только сбор показаний датчиков, но и управление какими-либо устройствами, или вывод сигналов, например, в активных способах измерения, предусматривающих излучение зондирующих сигналов, к аппаратному блоку подключаются выходные исполнительные устройства или преобразователи.

Таблица 5.1.

Программные компоненты инструментария **Data Acquisition Toolbox**

Компоненты	Назначение
m-файлы	Создают объекты устройств, собирают или выводят данные, конфигурируют значения параметров, оценивают состояние ресурсов, процесса измерений и регистрации
Движок	Сохраняет объекты устройств и значения их параметров, управляет записью зарегистрированных или выводимых данных, синхронизирует события
Адаптеры (интерфейсные программные модули)	Осуществляют двусторонний обмен информационными потоками параметров, данных, событий между движком и драйверами аппаратных блоков измерительной системы

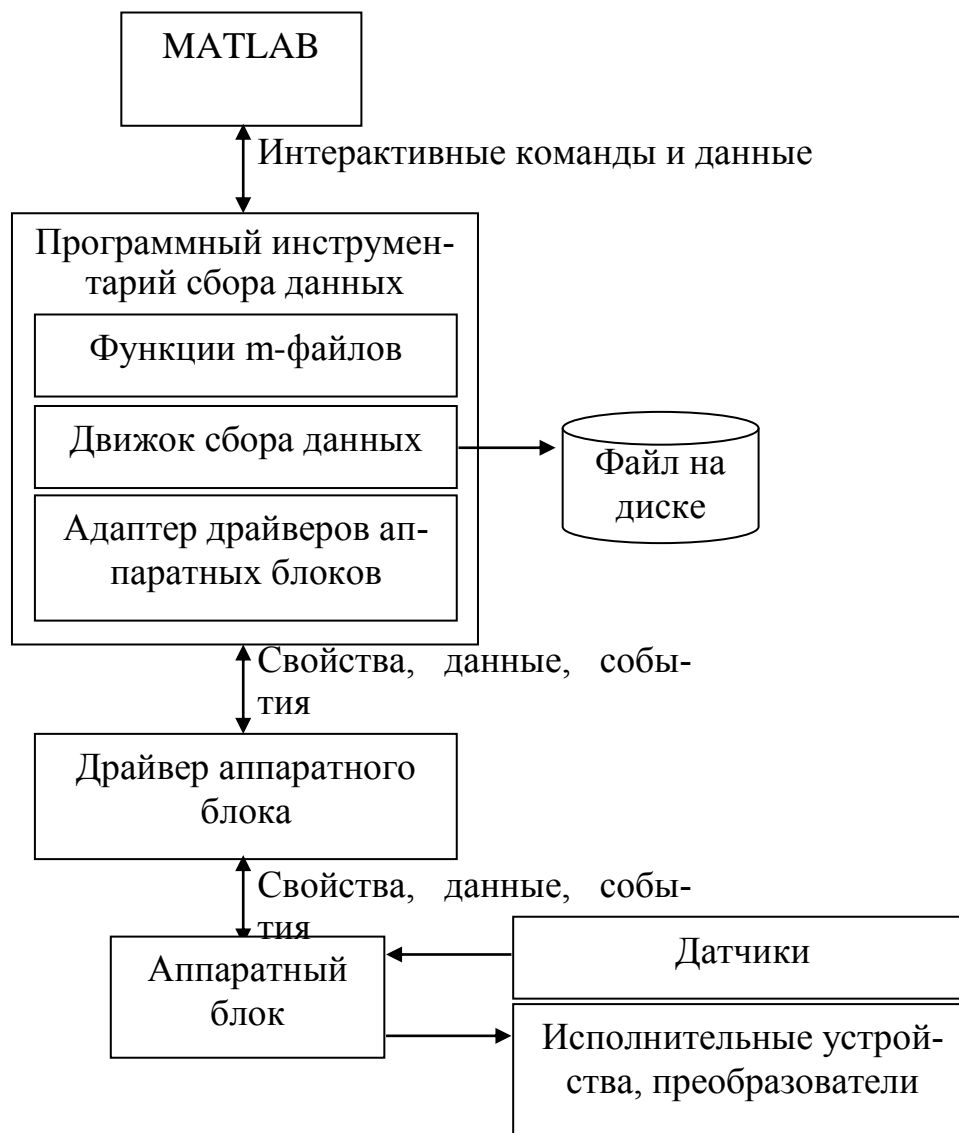


Рис. 5.4. Программно-аппаратная структура компьютерной измерительной системы



го ввода-вывода здесь отсутствует. Программирование объектов вызывает соответствующие действия в аппаратной части.

### Последовательность операций при регистрации сигналов

Процедура регистрации сигналов состоит из следующих этапов:

- 1) создание объекта устройства;
- 2) добавление каналов или линий к объекту устройства;
- 3) задание параметров регистрации, определяющих работу устройства;
- 4) ввод (AI) или вывод (AO) данных;
- 5) удаление объекта устройства (очистка).

Рассмотрим эти этапы более подробно.

#### *1. Создание объекта устройства*

Для создания объекта устройства необходимо вызвать соответствующую функцию (конструктор). Эти функции имеют свои названия в соответствии с создаваемыми объектами устройств.

Табл. 5.2.  
Функции создания объекта

Подсистема	Функция создания объекта
Аналоговый ввод	<code>analoginput('adaptor',ID);</code>
Аналоговый вывод	<code>analogoutput('adaptor',ID);</code>
Цифровой ввод/вывод	<code>digitalio('adaptor',ID);</code>

В этой таблице 'adaptor' - имя адаптера драйверов аппаратной части, для звуковой карты это **winsound**, ID – идентификатор аппаратного устройства, это необязательный аргумент, его можно не указывать, если используется звуковая карта с номером 0. Заметим, что кроме звуковых карт существует достаточно большое количество аппаратных модулей аналогового и цифрового ввода/вывода, выпускаемых различными фирмами. Эти модули здесь отдельно не рассматриваются.

Например, создание объекта аналогового ввода **ai** для звуковой карты с номером 0 осуществляется следующей строкой:

```
ai = analoginput('winsound');
```

#### *2. Добавление каналов или линий*

Прежде чем использовать созданный объект, к нему следует добавить, по крайней мере, один канал. Это осуществляется с помощью функции **addchannel**. Например, добавление двух каналов к объекту **ai** осуществляется следующим образом:

```
chans = addchannel(ai,1:2);
```

Объект устройства может быть представлен в виде контейнера каналов или линий, добавляемые каналы - в виде группы каналов, а линии – в виде группы линий. Соотношение между объектом аналогового ввода и каналами, которые он содержит, иллюстрируется схемой на рис. 5.7.

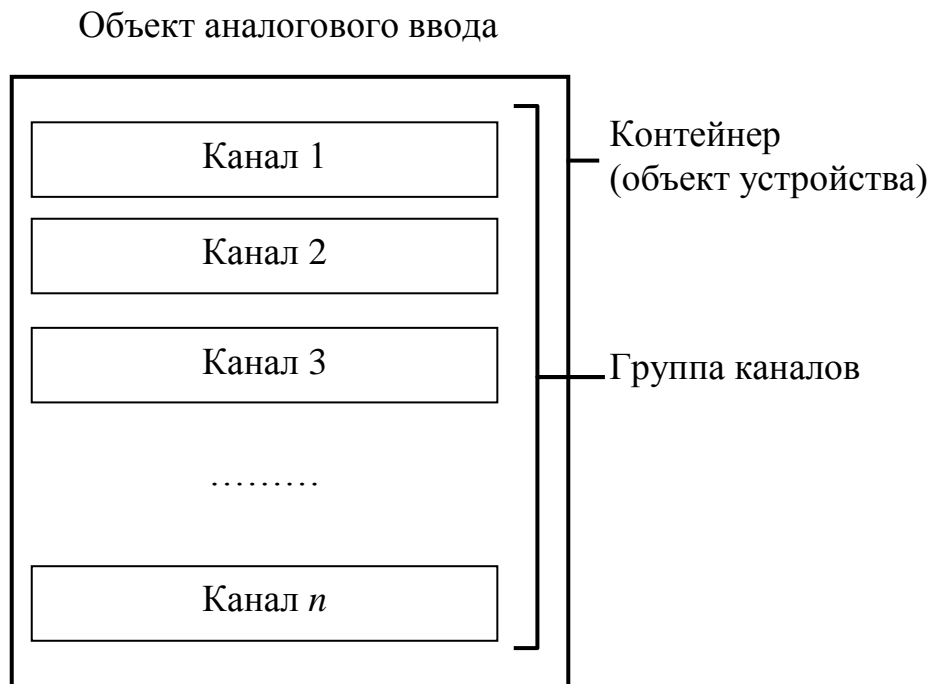


Рис. 5.7. Схема соотношения между объектом аналогового ввода и каналами, которые он содержит

Для объектов цифрового ввода/вывода схема выглядит аналогично.

### 3. Задание параметров регистрации, определяющих работу устройства

Управление работой аппаратного устройства ввода/вывода сигналов осуществляется заданием параметров его конфигурации. Это осуществляется в соответствии со следующими правилами:

- имена параметров нечувствительны к регистру (например, можно написать **set**, а можно и **Set**);
- имена параметров свойств можно сокращать;
- **set(ai)** задает все устанавливаемые параметры объекта **ai**, а **set(ai.Channel(index))** - все устанавливаемые параметры канала с номером **index**;
- **get(ai)** возвращает (позволяет получить) все устанавливаемые параметры объекта **ai**, а **set(ai.Channel(index))** - все устанавливаемые параметры канала с номером **index**.

### Типы параметров

Различают два типа параметров:

- общие параметры, устанавливаемые одинаковыми для всех каналов или линий;
- параметры каналов/линий, устанавливаемые для каждого канала или линии индивидуально.

Эти общие и индивидуальные параметры в свою очередь подразделяются также на два типа:

- базовые параметры, которые устанавливаются независимо от применяемых аппаратных средств ввода/вывода;
- параметры, специфические для каждого типа применяемого оборудования, при этом функции **set** и **get** при их использовании в программе выводят значения базовых параметров, после которых следуют значения параметров, специфические для данного вида аппаратного модуля.

### Синтаксис задаваемых параметров

Величины параметров могут быть заданы или получены (возвращены) тремя различными способами: с помощью функций **put** и **get**, точечной нотацией, именной индексацией.

#### *1 способ*

Синтаксис функций **put** и **get** может быть продемонстрирован следующими примерами:

```
out = get(ai,'SampleRate');
```

передает в переменную **out** значение частоты дискретизации **SampleRate**, установленное ранее в объекте **ai**;

```
set(ai,'SampleRate',11025); %
```

устанавливает частоту дискретизации аналого-цифрового преобразования, равной 11025 Гц.

#### *2 способ*

Упомянутые в предыдущем способе строки программного кода в точечной нотации имеет следующий синтаксис:

```
out = ai.SampleRate;  
ai.SampleRate = 11025;
```

Здесь в первой строке программного кода читается и передается в переменную **out** значение частоты дискретизации объекта **ai**, а во второй строке значение этого параметра задается равным 11025 Гц.



### *3 способ*

Именная индексация позволяет ассоциировать заданное имя с определенным каналом или линией. В примере, приведенном ниже, в первой строке имя **Chan1** связывается с первым каналом объекта **ai**, во второй строке осуществляется чтение значений диапазона первого канала аппаратного модуля и их передача в переменную **out**, а в третьей строке задаются значения диапазона первого канала в пределах от 0 до 10 единиц.

```
set(ai.Channel(1),'ChannelName','Chan1');  
out = ai.Chan1.UnitsRange;  
ai.Chan1.UnitsRange = [0 10];
```

### *4. Ввод/вывод данных*

Рассмотрим теперь порядок ввода/вывода сигналов. Здесь можно выделить три этапа:

- 1) старт объекта аппаратного устройства (модуля);
- 2) регистрация или передача данных;
- 3) остановка объекта устройства (модуля).

Рассмотрим эти этапы подробнее.

#### *4.1. Старт объекта устройства*

Для осуществления старта объекта, а вместе с ним и аппаратного устройства, соответствующего этому объекту, используется функция **start**, например

```
start(ai);
```

После того, как произведен старт объекта, автоматически **устанавливаются** его параметры **Running (AI)** или **Sending (AO)**.

Смысл слова «устанавливаются» в данном случае соответствует смыслу слова «включаются», противоположным значением обладает слово «сбрасываются», которое имеет смысл «выключаются».

#### *4.2. Запуск регистрации сигналов или данных*

Для ввода данных от датчика в программный движок или их запись в файл на диск в случае **AI**, а также вывода данных из движка на выход аппаратного модуля в случае **AO**, необходимо осуществить запуск, называемый в англоязычной литературе **trigger**. Существует несколько типов запуска на регистрацию или вывод данных, они приведены в таблице.

Тип запуска	Описание
Непосредственный	Запуск происходит сразу после выполнения функции <b>start</b> . Этот режим, как правило, установлен по умолчанию
Ручной	Запуск происходит только после того, как вручную введена функция <b>trigger</b>
Программный	Существует только для режима аналогового ввода сигналов <b>AI</b> . Запуск происходит тогда, когда выполняется заданное условие, например, величина входного сигнала превосходит заданный порог. При многоканальной регистрации можно задать номер канала, который используется для запуска регистрации

После того, как произошел запуск (**trigger**), параметры **Logging (AI)** или **Sending (AO)** автоматически устанавливаются.

#### 4.3. Остановка объекта устройства

Объект устройства останавливается в одном из трех случаев:

- когда получено (AI) или выведено (AO) заданное количество отсчетов;
- когда произошла просрочка, т. е. закончился заданный интервал времени (**time-out**);
- когда вручную введена функция **stop**

`stop(ai)`

#### Управление данными

##### *Предварительный просмотр данных в процессе регистрации*

Когда запущен объект аналогового ввода (AI), имеется возможность просматривать регистрируемые данные с помощью функции **peekdata**. Например, извлечь в переменную **out** 1000 последних отсчетов объекта **ai** можно с помощью следующего программного кода

```
out = peekdata(ai,1000);
```

При этом эти данные не извлекаются из движка. После извлечения данных они могут быть выведены в виде таблицы или графика.

##### *Извлечение данных в процессе регистрации*

В любой момент регистрации данных объектом AI данные могут быть извлечены из движка с помощью функции **getdata**. Например, извлечение в

переменную **out** 1000 отсчетов из объекта **ai** осуществляется следующим образом:

```
out = getdata(ai,1000);
```

Функция **getdata** передает управление выполнением программы системе **MATLAB** только тогда, когда все указанное количество отсчетов получено.

#### *5. Удаление и очистка объекта устройства*

Функция **delete** удаляет заданный объект устройства из движка, но не из рабочего пространства **MATLAB**

```
delete(ai)
```

При этом объект **ai** остается в рабочем пространстве **MATLAB**, но он не является действующим объектом, поскольку уже не связан с конкретным аппаратным устройством. Окончательно этот объект может быть удален командой **clear**

```
clear ai
```

Если же вначале очистить действующий объект устройства, то он больше не будет существовать в рабочем пространстве, но все еще будет находиться в движке. Его можно извлечь оттуда и восстановить с помощью функции **daqfind** следующим образом:

```
out = daqfind;  
ai = out(1);
```

В ряде случаев возникает необходимость сохранить объект, он сохраняется в **mat**-файле с помощью команды **save** следующим образом:

```
save ai
```

Объект загружается из файла с помощью команды **load**

```
load ai
```

Объект может быть конвертирован в эквивалентный код **MATLAB** с помощью функции **obj2code**

```
obj2code(ai,'ai_save')
```

Объект устройства может быть создан заново следующим образом:

```
ai = ai_save
```

Данные, зарегистрированные с помощью объекта AI, а также события, объекты устройств, информация об аппаратных модулях могут быть сохранены на диске, как показано в следующем примере:

```
set(ai,'LoggingMode','Disk&Memory')  
set(ai,'LogFileName','data.daq')  
set(ai,'LogToDiskMode','Index')
```

Информация из имеющегося файла может быть получена с помощью функции **daqread**. Для чтения всех данных используется код

```
data = daqread('data.daq');
```

Если необходимо загрузить из внешнего файла только объект и информацию об аппаратной части, можно использовать код

```
daqinfo = daqread('data.daq','info');
```

Далее проиллюстрируем данные положения на примере записи сигналов с помощью звуковой карты.

### 5.2.3. Непрерывная регистрация данных

Рассмотрим программные функции, применяемые для ввода и регистрации аналоговых сигналов. Знакомство с ними проиллюстрируем на примере задачи записи в течение одной секунды сигналов, подаваемых на два входа звуковой карты, и последующего вывода зарегистрированных данных в виде графиков зависимости входного напряжения от времени с записью данных в файл. Это непрерывная регистрация.

Далее на примерах будем показывать, как использовать эти функции для решения поставленной задачи. Эти места в тексте будут отмечены двойной жирной линией слева. При практической проработке этого материала текст программы следует набирать в редакторе системы MATLAB и после ввода каждый раз контролировать ее выполнение. Освоение проводим по пунктам в следующей последовательности.

1. Для того, чтобы можно было откомпилировать программу и запускать отдельно от системы MATLAB, ее следует оформить в виде программы-функции. Это делается с помощью записи в формате

```
function FileName
```

где FileName – имя файла. Будем считать, что в нашем случае программа называется Sound01, поэтому в первой строке следует написать

```
function Sound01
```

2. Функция **analoginput** конструирует объект аналогового ввода сигналов. Формат

```
AI = ANALOGINPUT('ADAPTOR'),
```

или

```
AI = ANALOGINPUT('ADAPTOR',ID)
```

Здесь ADAPTOR – название устройства, через которое осуществляется ввод аналоговых сигналов. Упомянем здесь только стандартную звуковую карту компьютера, для которой используется обозначение **winsound**. Вторая форма используется, если в компьютере установлено несколько таких устройств, при этом каждое из них имеет свой идентификатор ID. Если же звуковая карта одна, то идентификатор ID может быть опущен.

```
%Создать объект устройства ai ввода аналоговых сигналов в звуковую карту
```

```
ai = analoginput('winsound');
```

Если не поставить точку с запятой в конце этой строки, то будут выведены значения параметров звуковой карты, присваиваемые по умолчанию. В данном случае они имеют следующие значения:

- частота дискретизации 8000 Гц (отсчетов в секунду) по каждому каналу;
- длительность регистрации 1 с после старта;
- запись сигналов в переменную 'Memory', размещенную в оперативной памяти;
- один непосредственный запуск после старта программы.

3. После создания объекта, осуществляющего ввод аналоговых сигналов в компьютер, к нему необходимо добавить входные каналы. Это осуществляется с помощью функции **addchannel**.

Формат

```
CHANS = ADDCHANNEL(OBJ, HWCH)
```

или

```
CHANS = ADDCHANNEL(OBJ, HWCH, NAMES)
```

где OBJ – объект ввода аналоговых сигналов; HWCH – вектор номеров каналов, добавляемых к объекту, NAMES – имена каналов в виде вектора символьных переменных, присваиваемых номерам в HWCH, при этом количество элементов в NAMES должно равняться количеству элементов в векторе HWCH. Массив номеров каналов возвращается в переменную CHANS. Например, выражение

```
chan = addchannel(AI, [1 2]);
```

прибавляет два входных канала с номерами 1 и 2 к объекту AI и записывает в переменную **chan** их значения. При этом принято, что левый канал имеет номер 1, а правый – номер 2. Если нужно производит регистрацию по одному из каналов, следует просто указать его номер (1 или 2). Эти номера могут быть указаны и другим способом, которым мы воспользуемся для построения нашей программы. При этом номера этих каналов не будем записывать в какую-либо переменную.

```
%Добавление двух каналов с номерами 1 и 2 к объекту ввода аналоговых сигналов
```

```
addchannel(ai,1:2);
```

4. Конфигурирование свойств звуковой карты. Эта операция осуществляется функцией `set`, в простейшем случае имеющей формат

```
SET(H,'PropertyName',PropertyValue)
```

Здесь `H` – вектор параметров, которые будут устанавливаться для всех объектов; `PropertyName` – название параметра, задается в виде строковой переменной; `PropertyValue` – значение параметра.

В нашем случае зададим с помощью этой функцией частоту дискретизации звуковой карты. Как указывалось ранее, по умолчанию она равна 8000 Гц,

```
%По каждому каналу устанавливаем частоту дискретизации, обозначаемой как 'SampleRate' и равной 44,1 кГц, а длительность регистрации – одной секунде, при этом количество отсчетов на один запуск 'SamplesPerTrigger' будет равно 44100.
```

```
set(ai,'SampleRate',44100)  
set(ai,'SamplesPerTrigger',44100)
```

5. Запуск таймера, формирующего интервал времени регистрации. Формат

```
START(OBJ)
```

Эта функция запускает объект таймера с именем OBJ. Остановка таймера происходит по одному из условий:

- количество внутренних вызовов функции таймера превышает заданное значение;
- выполняется команда `STOP(OBJ)`;
- во время выполнения функции `START` происходит ошибка.

В нашем случае внутренние вызовы осуществляются с частотой дискретизации и общее количество таких вызовов 44100.

```
% Запуск системы на регистрацию  
start(ai)
```

После регистрации всех отсчетов *ai* происходит ее автоматическая остановка.

6. Извлечение из *ai* массива зарегистрированных данных с помощью функции **getdata**, которая возвращает массив данных из движка в рабочую область MATLAB и в простейшем случае имеет формат

```
DATA = GETDATA(OBJ)
```

дает количество зарегистрированных отсчетов, заданных ранее параметром *SamplesPerTrigger*. *DATA* это матрица размером  $M \times N$ , где  $M$  – количество отсчетов,  $N$  – количество каналов в объекте *OBJ*, по которым ведется регистрация.

Если необходимо получить значения моментов времени, в которые производится регистрация каждого отсчета, то используется функция *getdata* в следующем формате

```
[DATA, TIME] = GETDATA(OBJ)
```

Такая запись позволяет получить зарегистрированные отсчеты в заданном ранее количестве в матрице *DATA*, а также матрицу *TIME* размером  $M \times 1$ , где  $M$  - количество отсчетов, в которой содержатся значения времени, соответствующие каждому отсчету.

Для прерывания регистрации следует нажать сочетание клавиш **^C** (Control-C), при этом регистрация закончится, и управление будет передано в MATLAB. В нашем случае это будет выглядеть так:

```
% Получение зарегистрированных данных в переменную data  
[data,time] = getdata(ai);
```

При отсутствии точки с запятой в конце строки обе матрицы будут выведены на экран, сначала **data**, имеющая в нашем случае два столбца, а затем **time**, в которой содержатся данные в виде одного столбца.

7. Построение двумерных графиков зарегистрированных данных с помощью функции **plot**. Она имеет формат

```
PLOT(X,Y)
```

и осуществляет построение значений вектора  $Y$  в зависимости от значений вектора  $X$  по отрезкам, соединяющим точки. Если  $X$  и  $Y$  матрицы, график строится в зависимости от строки или колонки матрицы.

PLOT( $Y$ ) строит график  $Y$  в зависимости от номера строки, а различные типы линий, символов и их цветов могут быть построены с помощью функции PLOT( $X,Y,S$ ). Более подробное описание можно получить, вызвав справку.

```
%Построение графиков зарегистрированных данных  
plot(time,data)
```

8. Подготовка данных для записи. Зарегистрированные данные необходимо записать в файл для дальнейшей обработки. Как правило, запись осуществляется по колонкам, в первую из которых записываются моменты времени, в которые производятся отсчеты, а в последующие – зарегистрированные значения. Колонки друг от друга отделяются разделителями, в качестве которых обычно берутся пробел, запятая или табуляция, а строки отделяются символами **LF (Line Feed)**, что означает перевод строки, которые могут дополнительно сопровождаться символами **CR (Carriage Return)**, означающими возврат печатающей каретки. Поскольку в нашем примере время и данные записаны в разных переменных (матрицах), их надо свести в одну. Это можно сделать операцией конкатенации (объединения) матриц, в результате которой образуется новая матрица. Для этого в MATLAB применяются квадратные скобки. Выражение  $C = [A \ B]$  объединяет матрицы горизонтально, т. е. к последнему столбцу матрицы  $A$  пристыковывается первый столбец матрицы  $B$ . В свою очередь выражение  $C = [A; B]$ , содержащее точку с запятой, объединяет матрицы вертикально, т. е. к последней строке матрицы  $A$  снизу пристыковывается первая строка матрицы  $B$ .

```
%Объединение матриц  
out=[time data];
```

В нашем случае здесь образуется матрица **out**, содержащая три колонки, в первой из которых записаны отсчеты времени, во второй и третьей показания по первому и второму каналам звуковой карты соответственно. Всего строк в ней 44100.

9. Запись зарегистрированных данных в файл. Она осуществляется функцией записи с разделителями **dlmwrite**. Используется несколько вариантов этой функции, мы будем использовать следующий формат вывода данных из матрицы  $M$  в файл

```
DLMWRITE('FILENAME',M,'ATTRIBUTE1','VALUE1','ATTRIBUTE2','V  
ALUE2'...)
```



где 'FILENAME' строковая переменная, означающая имя файла, 'ATTRIBUTE', 'VALUE1' – первые и последующие атрибуты и их значения. В нашем случае мы укажем атрибутами:

- 'precision' точность вывода 10 значащих цифр;
- 'delimiter' разделитель табуляция, обозначается символами \t;
- 'newline' перевод строки на персональном компьютере pc.

```
% Вывод зарегистрированных данных в файл myfile.txt  
fil='myfile.txt';  
dlmwrite(fil,out,'precision',10,'delimiter','\t','newline','pc');
```

10. Удаление объекта **ai** из памяти и из рабочего пространства MATLAB производится, если необходимость работы с ним отпала.

```
delete(ai)  
clear ai
```

Полный текст программы-функции с именем Sound01 имеет вид:

```
1 function Sound01  
2 ai = analoginput('winsound');  
3 addchannel(ai,1:2);  
4 set(ai,'SampleRate',44100)  
5 set(ai,'SamplesPerTrigger',44100)  
6 start(ai)  
7 [data,time] = getdata(ai);  
8 plot(time,data)  
9 out=[time data];  
10 fil='myfile.txt';  
11 dlmwrite(fil,out,'precision',10,'delimiter','\t','newline','pc')  
12 delete(ai)  
13 clear ai
```

Результатом работы программы являются график, представленный ниже на рис. 5.8, и файл **myfile.txt**, в котором записаны значения времени и входных сигналов.

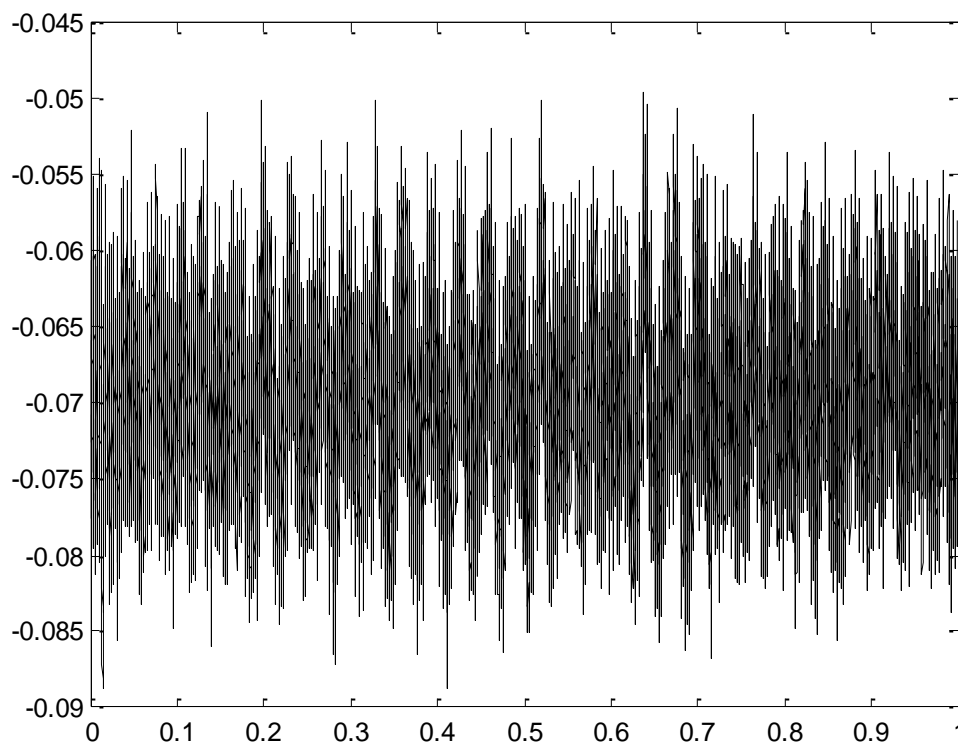


Рис. 5.8. Внешний вид графика шумов, зарегистрированного по двум каналам и выведенного в результате действия программы Sound01 без подключенных преобразователей

Поскольку на вход звуковой карты не подавались никакие сигналы, на графиках рис. 5.8 изображен собственный шум звуковой карты

#### 5.2.4. Регистрация сигнала и вывод спектра

Изменим текст программы, рассмотренной в п. 5.2.3., таким образом, чтобы выводился график спектра сигнала. Для расчета спектра используется функция FFT. Один из ее форматов имеет вид  $FFT(X)$ , где  $X$  – вектор. Для матриц это преобразование применяется ко всем столбцам. Эта функция осуществляет **прямое** дискретное преобразование Фурье в соответствии с выражениями

$$S(k) = \sum_{n=1}^N x(n) \cdot \exp\left[-\frac{j2\pi(k-1)(n-1)}{N}\right], \quad 1 \leq k \leq N. \quad (5.1)$$

и позволяет рассчитать комплексный спектр сигнала  $x(n)$ , заданного в виде отсчетов. Здесь  $S(k)$  – дискретный спектр в частотной области;  $x(n)$  – сигнал во временной области;  $N$  – количество отсчетов во временной и спектральной

областях;  $j = \sqrt{-1}$ ;  $k, n$  – номера отсчетов в спектральной и временной областях соответственно. Прямое преобразование Фурье позволяет рассчитать спектр сигнала по его реализации, зарегистрированной во временной области. Для расчета формы сигнала во временной области по известному спектру используется обратное преобразование Фурье, которое в дискретной форме имеет вид

$$x(n) = \frac{1}{N} \sum_{k=1}^N S(k) \cdot \exp\left[\frac{j2\pi(k-1)(n-1)}{N}\right], \quad 1 \leq n \leq N. \quad (5.2)$$

Обозначения здесь те же, что и для предыдущей формулы. Обратное дискретное преобразование Фурье осуществляется с помощью функции IFFT(X).

Здесь следует сделать одно очень важное замечание.

Название функции FFT происходит от первых букв словосочетания **Fast Fourier Transform**, что означает «быстрое преобразование Фурье». Алгоритм такого преобразования был разработан для ускорения вычислений. Он основан на последовательном делении всего массива чисел на 2 до тех пор, пока в каждой группе не оставалось бы по 1 элементу. Поэтому общее число отсчетов в исходном массиве должно равняться целой степени числа 2.

Прямое и обратное дискретное же преобразование Фурье, вычисляемое по формулам (5.1) и (5.2) соответственно, не требует обязательного выполнения последнего условия, и количество элементов в выборках может быть любым.

В системе MATLAB функция FFT в разных вариантах имеет следующий формат:

- FFT(X), где X – вектор отсчетов входного сигнала. Для матриц X эта операция применяется к каждой колонке.
- FFT(X,N) это N-точечное преобразование Фурье с добавлением нулей в конце, если длина входного вектора X меньше N и усечением до N, если эта длина больше N.

Существуют и другие варианты записи функции FFT, они здесь не рассматриваются.

Приведем текст программы регистрации сигнала и вывода данных в виде графика спектра.

```
function Sound02
% Создание объекта аналогового ввода, взаимодействующего
% с аппаратным устройством ввода сигналов (звуковой картой)
ai = analoginput('winsound');
addchannel(ai, 1);

% Конфигурирование объекта для регистрации
% в течение 2-х секунд данных с частотой
% оцифровки 8000 Гц.
```

```

Fs = 8000;
duration = 2;
set(ai, 'SampleRate', Fs);
set(ai, 'SamplesPerTrigger', duration*Fs);

% Старт регистрации и возвращение
% (передача в переменную data)
% зарегистрированных данных.
start(ai);
data = getdata(ai);

% Расчет спектральных компонент и
% построение графика спектра
% зарегистрированного сигнала data.
xfft = abs(fft(data));
mag = 20*log10(xfft);
mag = mag(1:end/2);
plot(mag);

% Очистка переменных
delete(ai);
clear ai;

```

### 5.2.5. Регистрация сигналов с запуском по превышению порога срабатывания

Часто приходится регистрировать сигналы, которые следуют в течение короткого времени с большими интервалами. Промежутки между такими сигналами регистрировать нет необходимости, чтобы не занимать больших объемов памяти. В этом случае производится регистрация по превышению порога, заданного заранее или меняющегося в процессе наблюдения. Примерами являются сигналы дискретной акустической эмиссии при разрушении материалов, а также сейсмосигналы, сопровождающие горные удары или землетрясения. Обобщенная форма таких сигналов приведена на рис. 5.9.

В процессе наблюдения выставляется порог срабатывания  $U_{п}$ . Его величина, с одной стороны, выбирается достаточно большой, чтобы не было ложных срабатываний от шума. С другой стороны, слишком высокий уровень приведет к пропуску сигналов небольшой амплитуды.

При отсутствии сигнала запись осуществляется в промежуточную область и не переносится в основную память.

При появлении сигнала в момент времени  $t_c$  происходит превышение порога, после чего происходит регистрация сигнала в течение последующего промежутка времени  $\Delta t_{по}$ , называемого постисторией. Для правильного срабатывания необходимо установить направление изменения сигнала. **Rising** соответствует запуску при возрастании сигнала.

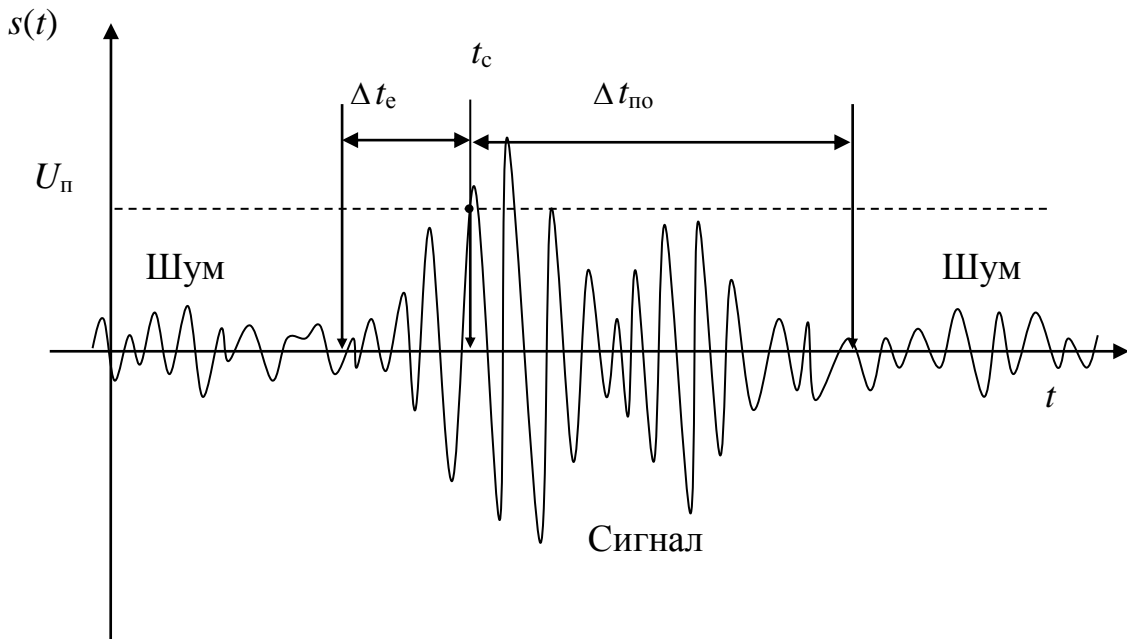


Рис. 5.9. Регистрация сигнала по превышению порога срабатывания

Как видно из рис. 5.9, при такой регистрации теряется начальная часть сигнала, соответствующая промежутку времени  $\Delta t_e$ , называемая предысторией (**pretriggering**). В то же время эта часть является наиболее важной, особенной в некоторых случаях. Именно начальная часть несет информацию об источнике возникновения упругих волн при разрушении материалов и образовании трещин. Конечная часть сигнала несет информацию о среде распространения. Кроме того, если регистрируются сигналы акустической эмиссии в образцах небольших размеров, то здесь будут сказываться отражения волн от стенок образца. Они полностью замаскируют действие начальной части, и ценная информация будет потеряна.

Для того, чтобы не потерять эту начальную часть, производится регистрация с предысторией. С этой целью заранее устанавливается время  $\Delta t_e$  или соответствующее ему количество отсчетов, которые переписываются в основную часть памяти при превышении порога и к которым будут дописываться либо заданное количество последующих отсчетов, либо такая запись будет производиться в течение заданного промежутка времени  $\Delta t_{по}$ .

С учетом вышесказанного рассмотрим программу регистрации с предысторией, расчетом спектра и выводом графиков сигнала и спектра. Там, где это необходимо, добавлены комментарии.

```
function Sound03
% Регистрация сигнала с предысторией,
% расчетом спектра и выводом графиков
% сигнала и спектра
```

```

% Создание объекта
AIVoice = analoginput('winsound');
% Регистрация по 1 каналу
chan = addchannel(AIVoice,1);
% Регистрация в течение двух секунд
duration = 2;
mytime = 0.1;
% Частота дискретизации
SR = 44100;
% Время регистрации предыстории
PreTr = 0.01;
% Установка частоты дискретизации
set(AIVoice,'SampleRate',SR)
% Проверка частоты дискретизации
ActualRate = get(AIVoice,'SampleRate');
% Установка количества отсчетов постистории
set(AIVoice,'SamplesPerTrigger',ActualRate*duration)
% Установка 1 канала в качестве канала запуска
set(AIVoice,'TriggerChannel',chan)
% Установка программного режима запуска
set(AIVoice,'TriggerType','Software')
% Запуск по возрастанию сигнала
set(AIVoice,'TriggerCondition','Rising')
% Установка порога запуска 0,05 В
set(AIVoice,'TriggerConditionValue',0.05)
% Установка количества отсчетов предыстории
set(AIVoice,'TriggerDelayUnits','Samples')
set(AIVoice,'TriggerDelay',-ActualRate*PreTr)
start(AIVoice)
[data,time] = getdata(AIVoice,mytime*ActualRate);
M = mean(data);
data = data-M;
subplot(2,1,1)
plot(time,data)
xlabel('Время, с')
ylabel('Сигнал, В')
axis([-PreTr mytime-PreTr -1.05 1.05]);
grid on
% Determine the frequency components of the data.
[m,n] = size(data);
T = time([m])+PreTr;
df = 1/T;
xfft = abs(fft(data));
mag = 20*log10(xfft);
mag = mag(1:end/2);

```

```

[m,n] = size(mag);
fi = 1:m;
f = df*fi;
subplot(2,1,2)
plot(f,mag);
xlabel('Частота, Гц')
ylabel('Уровень, В')
axis([0 ActualRate/2 -50 50]);
grid on
wait(AIVoice,2)
delete(AIVoice)
clear AIVoice

```

При запуске этой программы следует учесть, что если за время регистрации не приходит сигнал, выдается сообщение об ошибке.

### 5.3. Компиляция программ

В некоторых случаях необходимо осуществлять работу написанной программы без запуска системы MATLAB. В этом случае следует произвести компиляцию текста программы и получить загружаемый файл, который можно будет запускать непосредственно под действием операционной системы. Для этого используется команда

```
mcc -m filename.m
```

где filename.m – имя компилируемого файла. В нашем случае эта команда имеет вид

```
mcc -m Sound01.m
```

Здесь Sound01.m – имя файла, в котором содержится созданная функция Sound01.

Результатом такой операции будет создание исполняемого файла Sound01.exe, который можно будет запускать самостоятельно, даже если на компьютере не установлена систем MATLAB.

## КОНТРОЛЬНЫЕ ВОПРОСЫ К ГЛАВЕ 5

1. Каким образом можно осуществить ввод сигналов от аналоговых датчиков в компьютер?
2. В чем состоит назначение мультиплексора при вводе аналоговых сигналов в компьютер?
3. Каким образом следует выбирать частоту дискретизации при аналого-цифровом преобразовании?
4. Какие параметры следует учитывать при проектировании регистрации аналоговых сигналов с помощью компьютера?
5. Опишите последовательность действий при подготовке эксперимента с помощью компьютера.
6. Что такое драйвер? Каково его назначение при работе с внешними устройствами?
7. Опишите функциональную схему звуковой карты. Как с ее помощью можно регистрировать сигналы датчиков?
8. Что такое объекты и классы в компьютерной программе?
9. Опишите программные компоненты инструментария Data Acquisition Toolbox, в чем состоит их назначение, как они взаимодействуют с операционной системой?
10. Опишите этапы процедуры регистрации сигналов.
11. Какие виды запуска используются при регистрации сигналов на компьютерах?
12. Перечислите программные функции, используемые при непрерывной регистрации сигналов.
13. Какие параметры звуковой карты или других АЦП устанавливаются при подготовке эксперимента перед его началом?
14. Прокомментируйте программу ввода аналоговых сигналов, что происходит при выполнении каждой строки программного кода?
15. Какие функции и каким образом нужно использовать в программе, чтобы рассчитать и вывести на экран графики спектров сигналов?
16. Для чего применяется запуск регистрации по превышению порога, какие параметры должны при этом учитываться и устанавливаться перед началом регистрации?
17. Для чего нужна регистрация предыстории при запуске по превышению порога?
18. Что такое компиляция программ, с какой целью она используется и каким образом осуществляется?



## ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

- А**
- аргумент
    - входной, 23
    - выходной, 23
- В**
- валидация, 74
  - вектор, 10
    - размер, 10
  - вычисление
    - загрузочный модуль, 13
    - из редактора, 13
    - прямое, 12
- Г**
- графика
    - векторная, 30
    - растровая, 32
- Д**
- деление
    - справа налево, 23
  - драйвер, 79
- И**
- изображение, 52
    - алгоритмы сжатия, 30
    - бинарное, 33
    - глубина цвета, 29
    - графические режимы, 30
    - информация о, 61
    - контрастность, 59
    - модель, 28
    - модель RGB, 29
    - обработка, 57
    - размер, 28
    - серое, 33
    - способность разрешающая, 28, 33
    - формат файла, 29
    - цветное, 33
    - цифровое, 28
    - чтение и запись, 57
- инструментарий**
- Data Acquisition Toolbox, 83
  - компоненты, 83
- исследование**
- научное, 7
  - прикладное, 7
  - фундаментальное, 7
- К**
- карта
    - звуковая, 81
  - компоненты
    - соединенные, 70
  - константа, 15
    - переменная системная, 15
    - символьная, 16
    - числовая, 15
- М**
- массив
    - размерность, 10
  - матрица, 10
    - магическая, 21
    - разреженная, 10
  - минус
    - унарный, 14
  - моделирование, 7
  - модель, 7
    - компьютерная, 8
    - математическая, 7
    - физическая, 7
- О**
- объект
    - программный, 85
  - окно
    - командное, 11
  - операнд, 22
  - оператор, 22
    - двоеточие, 16
- П**
- параметр

входной, 23  
выходной, 23  
переменная  
индексированная, 11, 19  
ранжированная, 17  
пиксель, 28  
подграфик, 48  
преобразователь  
чувствительность, 80  
программа  
компиляция, 103  
программирование  
класс, 82  
объект, 82  
объектно-ориентированное, 82

## Р

регистрация  
запуск, 90  
непрерывная, 92  
по порогу, 100  
предыстория, 101

## С

сигнал  
спектр, 98  
сплайн, 31  
способность  
разрешающая, 55

## Ф

файл  
имя, 11  
расширение, 11  
функция, 23  
m-функция, 23  
ввода-вывода, 26  
внешняя, 23  
возврат результата, 23  
встроенная, 23

## Э

элемент  
структурный, 70

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Глоссарий.ru. [Электронный документ] (<http://www.glossary.ru/>). Проверено 05.11.2009.
2. Лычкина Н.Н. Современные тенденции в имитационном моделировании. – Вестник университета, серия Информационные системы управления, №2. – М., ГУУ., 2000 г.
3. MATLAB Getting Started Guide. The Mathworks Company. 2009, 108 pp. [Электронный документ] getstart.pdf, [http://www.mathworks.com/academia/student\\_center/tutorials/launchpad.html](http://www.mathworks.com/academia/student_center/tutorials/launchpad.html)
4. Дьяконов В. П. MATLAB: учебный курс. – СПб.: Питер, 2001.
5. Дьяконов В. П., Абраменкова И. В. MATLAB 5. Система символьной математики. - М.: Нолидж, 1999.
6. Дьяконов В., Круглов В. Математические пакеты расширения MATLAB: Специальный справочник.-СПб.:Питер, 2001.- 480 с.
7. Чернышов Н.М., Коробкина Т.П. Особенности распределения и формы концентрирования платиноидов и золота в железистых кварцитах Лебединского месторождения КМА // Вестник Воронежского университета. Геология. 2005. № 1, с. 140-152.
8. daqquickref.pdf. 2008, 14 pp. [Электронный документ]
9. Data Acquisition Toolbox 2. User's Guide. 2009, 742 pp. . [Электронный документ] [http://www.mathworks.com/access/helpdesk/help/pdf\\_doc/daq/daqug.pdf](http://www.mathworks.com/access/helpdesk/help/pdf_doc/daq/daqug.pdf)