Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Л. А. АРТЮШИНА Е. А. ТРОИЦКАЯ

АНАЛИЗ ДАННЫХ В СРЕДЕ MATLAB

Учебно-практическое пособие

Владимир 2022

УДК 004.43
ББК 32.973
А86

Рецензенты:

Доктор технических наук, профессор
зав. кафедрой информационных систем и программной инженерии
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
И. Е. Жигалов

Доктор технических наук, профессор
зав. кафедрой информатики и защиты информации
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
М. Ю. Монахов

Кандидат физико-математических наук
доцент кафедры специальной техники и информационных технологий
Владимирского юридического института
Федеральной службы исполнения наказаний
А. В. Хорошева

Издается по решению редакционно-издательского совета ВлГУ

Артюшина, Л. А. Анализ данных в среде MATLAB : учеб.-практ. пособие / Л. А. Артюшина, Е. А. Троицкая ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2022. – 272 с. – ISBN 978-5-9984-1526-5.

А86

Рассмотрена технология анализа данных в среде MATLAB. Представлены примеры и иллюстрации, поясняющие теорию. Приведено более 200 упражнений, позволяющих проверить знания по всем рассматриваемым темам.

Предназначено для студентов вузов 3 – 4-го курсов всех форм обучения направлений подготовки 10.03.01 «Информационная безопасность», 10.05.04 «Информационно-аналитические системы» при проведении практических и лабораторных работ по дисциплине «Анализ данных», а также при изучении курсов, связанных с математическими расчетами.

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС ВО.

Ил. 86. Табл. 8. Библиогр.: 29 назв.

УДК 004.43
ББК 32.973

ISBN 978-5-9984-1526-5

ВВЕДЕНИЕ

Что такое MATLAB? Это широко применяемая разными компаниями универсальная профессиональная среда для различного рода расчетов и анализа данных.

Почему именно MATLAB? Потому что эта система позволяет выстраивать все рабочие процессы с данными в одной среде, без погружения в «глубины» программирования читать, преобразовывать и анализировать данные из различных источников.

В пособии последовательно представлены все этапы анализа данных: чтение, предобработка, собственно анализ и визуализация различных типов данных с применением системы MATLAB. Учебный материал разбит на главы, соответствующие логике изложения материала. Рассматриваются: основные типы данных MATLAB и программные средства для работы с ними; программные средства математических и символьных вычислений MATLAB; средства визуализации вычислений, импорт и экспорт данных; предобработка и анализ данных двумя способами – с помощью инструмента Live Editor и стандартным способом, т. е. путем написания программного кода.

Пособие содержит большое количество примеров, демонстрирующих и поясняющих каждый этап анализа данных. В главах предусмотрены практические работы, включающие в себя по 16 индивидуальных заданий.

Глава 1
ВВЕДЕНИЕ В MATLAB. РЕЖИМЫ РАБОТЫ MATLAB

1.1. Командный режим

После запуска системы MATLAB на экран выводится основное окно рабочей области среды, которое называется «рабочий стол» (рис. 1.1).

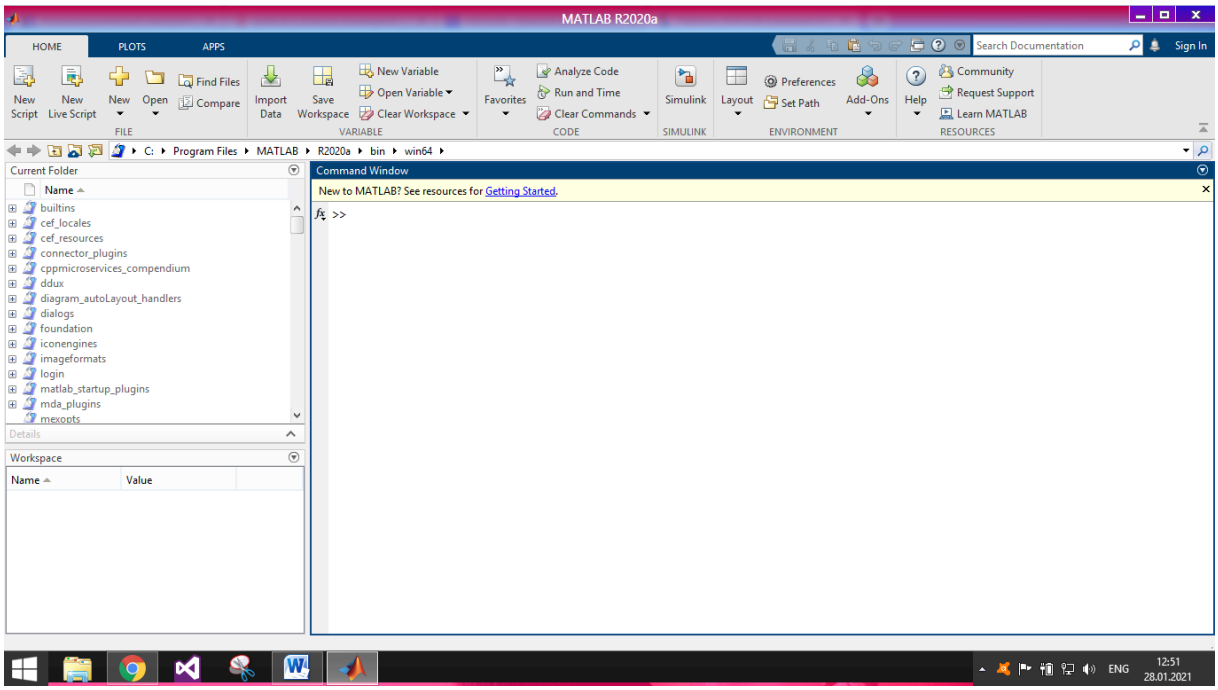


Рис. 1.1. Внешний вид окна «рабочий стол» системы MATLAB

Это окно содержит:

– строку заголовка;

– строку главного меню;

– панель инструментов с кнопками, позволяющими выполнить некоторые наиболее распространенные операции, которые можно выбрать и через меню.

Сразу под панелью инструментов находятся окна (слева направо):

– *Current Folder:* позволяет установить текущую папку;

– *Command Window* (окно команд) – это основная область, в которой команды можно вводить в командной строке. На это указывает символ «>>». Окно позволяет ввести отдельные операторы и про-

смотреть сгенерированные результаты. Для выполнения введенной команды необходимо нажать клавишу *Enter*. Пока она не нажата, вводимое выражение можно отредактировать. После нажатия *Enter* отобразить ранее набранные команды можно с помощью клавиш управления ↑ и ↓;

– браузер *Workspace* (рабочее пространство): позволяет просмотреть содержимое рабочей области в MATLAB и в интерактивном режиме управлять им. Когда это необходимо, для каждой переменной или объекта в рабочей области браузер может отобразить статистику, например минимум, максимум, среднее значение и т. д. Также непосредственно в браузере можно отредактировать содержимое скалярных переменных рабочей области. Для этого необходимо щелкнуть правой кнопкой мыши по переменной и выбрать редактор переменных *Edit Value*. Чтобы отредактировать в браузере другие переменные, достаточно дважды кликнуть имя переменной и открыть ее в редакторе переменных.

Рассмотрев основные элементы интерфейса, перейдем непосредственно к работе с MATLAB в режиме командной строки.

В качестве примера вычислим сумму векторов $\vec{a}(1;4)$ и $\vec{b}(-6;5)$.

В соответствии с определением операции сложения двух векторов их сумму можно записать в виде $\vec{c} = \vec{a} + \vec{b}$, где $\vec{c}$ – результирующий вектор, координаты которого равны суммам соответствующих координат векторов $\vec{a}$ и $\vec{b}$. Имеем $\vec{c}\ (1-6;4+5), \vec{c}(-5;9)$.

Текст программы в режиме командной строки:

```
>> a = [1 4];
>> b = [-6 5];
>> c=a+b
c =
   -5     9
```

1.2. Программный режим. Создание и выполнение скриптов

В режиме командной строки команда выполняется сразу же после ее ввода. Такой режим удобен, если нужно быстро проверить, как работает та или иная функция, или решить небольшую задачу.

Недостаток такого режима состоит в том, что при перезапуске среды MATLAB последовательность команд не сохраняется нигде, кроме окна истории, которое не предназначено для постоянного хранения последовательности команд.

Для больших задач, а также для задач, которые нужно запускать на выполнение несколько раз, удобнее сохранить последовательность действий в так называемый скрипт (script). Скрипты сохраняются в файлы с расширением .m, которые в документации иногда называются m-файлы (m-files).

Скрипты представляют собой обычные текстовые файлы, которые можно создавать и редактировать в любом текстовом редакторе, позволяющем сохранять текст без служебной информации, но в среде MATLAB уже есть встроенный текстовый редактор, специально предназначенный для программирования на языке MATLAB.

Для того чтобы создать скрипт и открыть его в редакторе MATLAB, нужно в главном меню нажать кнопку *New Script* на панели инструментов вкладки *Home* (рис. 1.2). При этом откроется окно редактора *Editor*, показанное на рис. 1.3.

Рис. 1.2. Создание скрипта

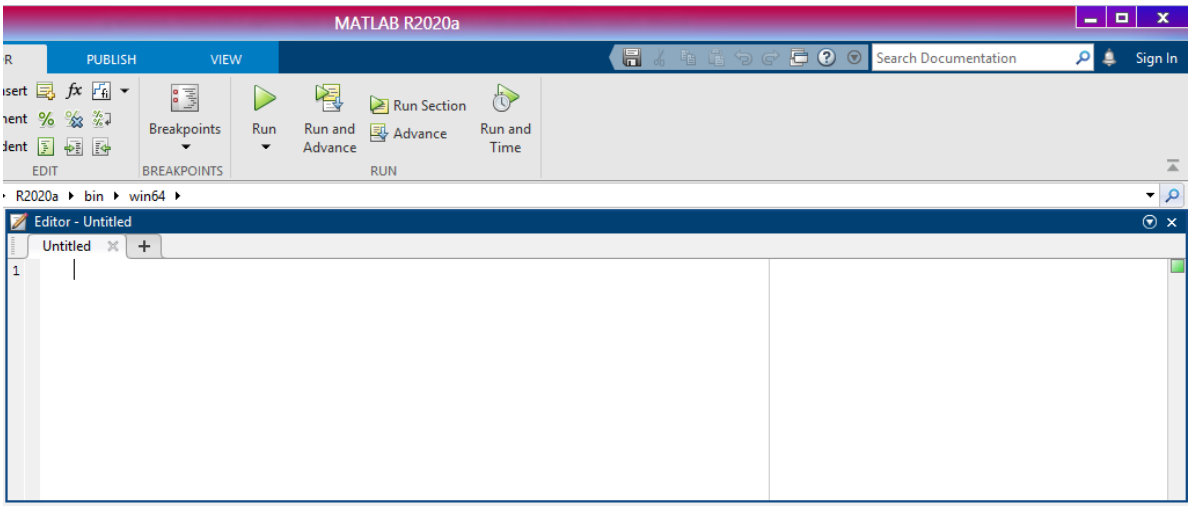


Рис. 1.3. Окно редактора *Editor*

Сохраним скрипт. Для этого на вкладке *Editor* выберем команду *Save*, в открывшемся диалоговом окне зададим имя скрипта, например exsample1.m (рис. 1.4, 1.5).

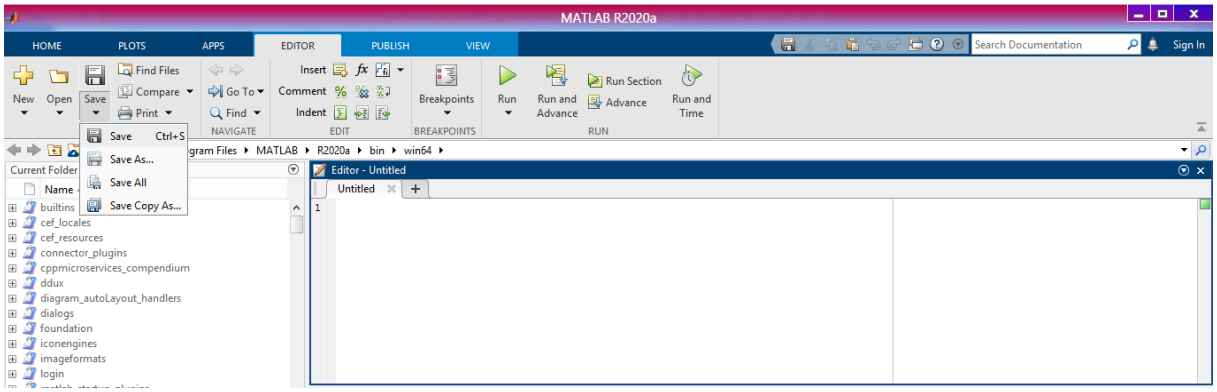


Рис. 1.4. Сохранение m-файла

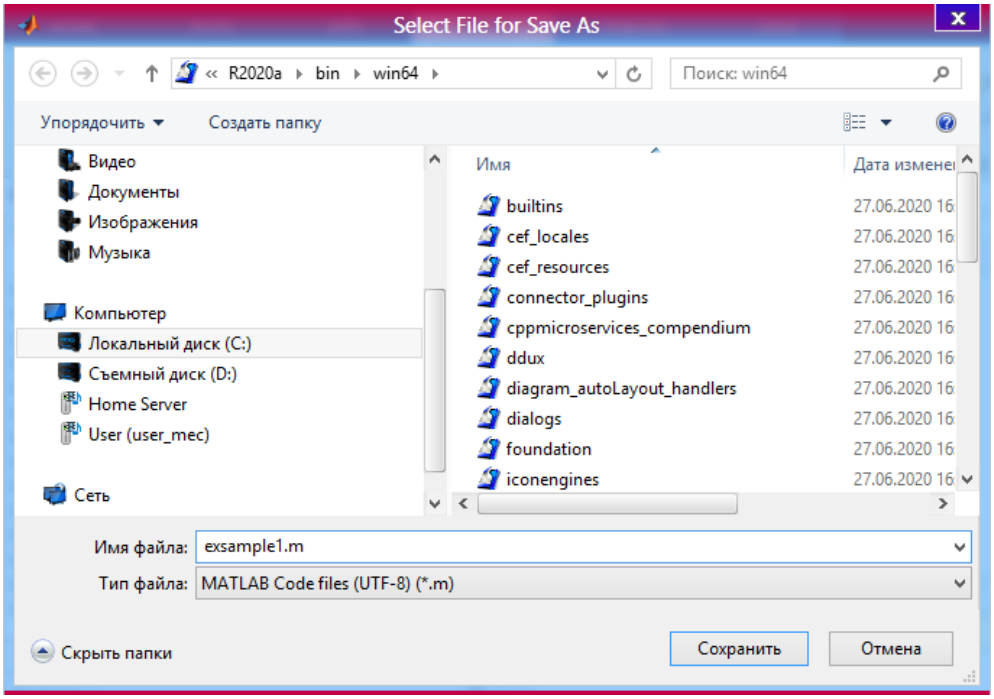


Рис. 1.5. Задание имени m-файла

Встроенный редактор *Editor* – довольно удобный инструмент, который заметно облегчает написание и исправление больших программ. Среди особенностей редактора можно отметить:

– раскраску синтаксиса языка MATLAB;
– подсветку парных скобок;
– подсказки параметров функций во всплывающем окне;

– наличие встроенного отладчика;

– подсветку ошибок по мере набора текста программы без запуска скрипта на выполнение (рис. 1.6).

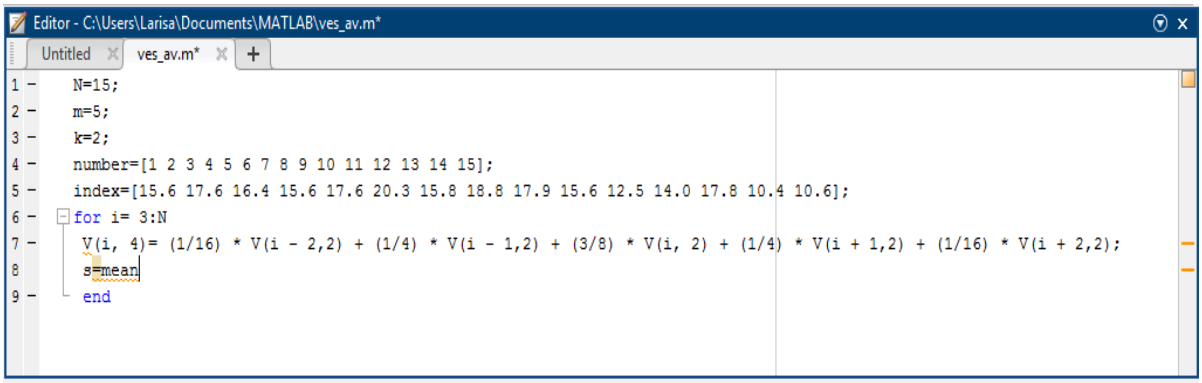


Рис. 1.6. Некоторые возможности редактора *Editor*

Для того чтобы запустить этот скрипт на выполнение, нужно выбрать команду *Run* во вкладке *Run*... (вместо многоточия будет указано имя файла скрипта), или нажать на одноименную кнопку на вкладке *Run*, или нажать горячую клавишу F5.

В качестве примера напишем в виде скрипта программу, вычисляющую сумму двух векторов (vector.m):

```
a = [1 4];
b = [-6 5];
c=a+b;
disp(c);
>> vector
   -5     9
```

Отличие от первой программы состоит в отсутствии символа «>>» и наличии команды **disp(c);**, выводящей значение суммы. Предполагается, что дальнейшие примеры запускаются в виде скрипта.

1.3. Версии MATLAB

Многие операторы и функции MATLAB для корректного сохранения данных требуют указания используемой пользователем версии MATLAB. Список поддерживаемых сегодня версий системы и их обозначения представлены в таблице.

Версии системы MATLAB

Значение версии	Загрузки в версиях MATLAB	Поддерживаемые функции	Сжатие	Максимальный размер каждой переменной
v7.3	7.3 (R2006b) или позже	Сохранение и загрузка частей переменных и всех функций версии 7. Версия 7.3 также поддерживает сохранение значений переменных без сжатия с помощью опции nocompression	Да (значение по умолчанию)	≥ 2 Гбайт на 64-битных компьютерах
v7	7.0 (R14) или позже	Кодировка символов Unicode, которая включает в себя совместный доступ к файлам между системами, использующими различные схемы кодировки символов. По умолчанию поддерживаются все функции версии 6. Версия 7 также поддерживает сохранение значений переменных без сжатия с помощью опции nocompression	Да (значение по умолчанию)	2^{31} байт на переменную
v6	5 (R8) или позже	N-мерные массивы, массивы ячеек, массивы структур, имена переменных длиной более 19 символов. Поддерживаются все функции версии 4	Нет	2^{31} байт на переменную
v4	Все	Двумерный double, символ и разреженные массивы	Нет	100 000 000 элементов на массив и 2^{31} байт на переменную

Вопросы для самоконтроля

1. В чем состоит различие режимов работы MATLAB?

2. Перечислите значения поддерживаемых на сегодняшний день версий системы MATLAB.

3. Кратко охарактеризуйте поддерживаемые каждой версией функции.

Глава 2
ТИПЫ ДАННЫХ MATLAB

Анализ данных начинается с выбора подходящих типов переменных для их хранения. Следующий шаг – преобразование так называемых плохих данных, т. е. содержащих пропуски, неотсортированных, разного формата и так далее, в «хорошие» данные, например приведенные к единому формату. Эти предварительные шаги гарантируют значимые заключения в последующих частях анализа.

Наиболее часто анализируются различные комбинации из числовых и символьных данных, которые для удобства преобразуют в табличный формат – универсальную форму представления разнотипных данных.

Учитывая сказанное, анализ данных начнем со знакомства с числами, строками и таблицами и их реализацией в системе MATLAB.

2.1. Типы данных

В системе MATLAB существуют 16 типов, или классов, данных. Так как MATLAB ориентирован на работу с массивами и матрицами, каждый из этих классов имеет форму матрицы или массива. Перечислим основные из них:

– числовые типы;
– символы и строки;
– даты и время;
– категориальные массивы;
– таблицы;
– расписания;
– структуры;
– массивы ячеек;
– указатели на функции;
– контейнеры Map;
– временные ряды.

В рамках пособия уделим внимание только тем типам данных, которые необходимы для выполнения практических работ. Указатели на функции и контейнеры Map не рассмотрим.

2.2. Числовые типы

В таблице представлены наиболее часто используемые в MATLAB числовые типы.

Числовые типы данных

Название типа данных	Форма типа данных	Область значений
double	Массивы с двойной точностью	Для отрицательных чисел: от $-1{,}79769 \cdot 10^{308}$ и до $-2{,}22507 \cdot 10^{-308}$ Для положительных чисел: от $2{,}22507 \cdot 10^{-308}$ и до $1{,}79769 \cdot 10^{308}$
single	Массивы с одинарной точностью	От $3{,}4 \cdot 10^{38}$ до $-3{,}4 \cdot 10^{38}$
int8	8-битные массивы целого числа со знаком	От -2^7 до $2^7 - 1$
int16	16-битные массивы целого числа со знаком	От -2^{15} до $2^{15} - 1$
int32	32-битные массивы целого числа со знаком	От -2^{31} до $2^{31} - 1$
int64	64-битные массивы целого числа со знаком	От -2^{63} до $2^{63} - 1$
uint8	8-битные массивы беззнаковых целых чисел	От 0 до $2^8 - 1$
uint16	16-битные массивы беззнаковых целых чисел	От 0 до $2^{16} - 1$
uint32	32-битные массивы беззнаковых целых чисел	От 0 до $2^{32} - 1$
uint64	64-битные массивы беззнаковых целых чисел	От 0 до $2^{64} - 1$

Как видно из таблицы, это целые числа со знаком и без знака, числа с плавающей запятой двойной и одинарной точности. По умолчанию MATLAB хранит все числовые значения в формате с плавающей запятой двойной точности.

ВАЖНО! Следует помнить, что вычисления с нецелыми числами будут неточными.

Возможно преобразование типов числовых данных. Для этого в MATLAB используется явное преобразование типов.

Пример. Преобразование числа двойной точности к формату одинарной точности:

```
>> a = 1.23;
>> b = single (a)
b =
  single
    1.2300
>> whos
  Name      Size            Bytes  Class     Attributes
  a         1x1                 8  double
  b         1x1                 4  single
```

Пример. Преобразование числа в формате двойной точности в 16-разрядное целое число со знаком, дробная часть числа меньше 0,5:

```
>> a = 1.23;
>> c=int16(a)
c =
  int16
   1
```

Пример. Преобразование числа в формате двойной точности в 16-разрядное целое число со знаком, дробная часть числа меньше или равна 0,5:

```
>> a=2.74;
>> c=int16(a)
c =
  int16
   3
```

Все числовые типы поддерживают основные операции над массивами, такие как индексация, изменение и математические операции.

Вопросы для самоконтроля

1. Назовите и кратко охарактеризуйте основные числовые типы данных.
2. Назовите основные диапазоны значений числовых типов данных.
3. Перечислите основные функции преобразования числовых форматов. Приведите примеры.

4. Какую важную для анализа данных информацию можно получить с помощью разных числовых типов данных? Перечислите их. Приведите примеры использования.

Практическая работа

ЧИСЛОВОЙ ТИП ДАННЫХ

Задание. Вычислите значения выражений. Оформите отчет.

Отчет по практической работе включает в себя:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий выполненным расчетам;

○ результат вычислений.

Варианты заданий

Вариант 1. $a = -1{,}3; b = 0{,}91; c = 0{,}75; x = 2{,}32; k = 8.$

$$y = \sin\frac{a-x}{c} + 10^{\sqrt[4]{\sqrt[3]{\frac{a-kx^2}{2b}+\frac{\cos kx^2}{\mathrm{tg}3}-\frac{bc}{ax}}}}.$$

Вариант 2. $k = 2; x = 0{,}32; d = 1{,}25; n = -4; b = 0{,}75; c = 2{,}2.$

$$y = 10^{-3}\mathrm{tg}kn - \frac{(x-d)(x^2+b^2)}{\sqrt[3]{x^2+b^2-cd}} - \frac{\cos kx}{\sin 5}.$$

Вариант 3. $i = 5; k = -2; x = 0{,}1; a = 25{,}2; b = 2{,}35.$

$$y = \mathrm{tg}ik - \frac{ax^3-b}{(a+b)^2} + 10^3 e^{-5} + \sqrt[3]{\frac{10^2|xk|}{(a+b)^2}}.$$

Вариант 4. $a = -1{,}25; c = 0{,}05; d = 2{,}5; i = 5; x = 1{,}35.$

$$y = \frac{\sqrt{|c-d|+(a+c)^2}}{\sin 2i} + 10^{-3}e^{ix} - \frac{|c-d|+a^2}{\sqrt[n]{(a+c)^2}}.$$

Вариант 5. $k = 2; x = -2{,}5; c = 0{,}31; a = 0{,}93; b = 5{,}61.$

$$y = \frac{\ln|kx|}{\sin 7} - \sqrt{x-a^2} - \frac{10^4 a - b}{\cos kx} + \sqrt[3]{x-a^2} + c^3 x.$$

Вариант 6. $k = -2; a = 3{,}5; b = 0{,}35; x = 1{,}523.$

$$y = 10^4\frac{ax}{b^2} - \left|\frac{a-b}{kx}\right| + \frac{\ln 3}{\sqrt[3]{ax+b^2}} - e^{-kx}.$$

Вариант 7. $a = 1{,}7$; $b = -1{,}25$; $c = -0{,}3$; $x = 2{,}5$; $k = 3$.

$$y = \sqrt{\frac{abc}{2{,}4}} - \frac{0{,}7abc}{\sin 7} + 10^4 \sqrt[5]{|\cos kx|} - \frac{|b-a|}{kx}.$$

Вариант 8. $a = 1{,}3$; $b = 2{,}42$; $c = 0{,}83$; $x = 1{,}5$; $k = 2$.

$$y = \frac{|a^2 - b^2|}{\sin kx} - \frac{k^2 + \operatorname{tg} 3k}{e^{kx}} - 10^4 \sqrt[5]{|\sin kx - bc|}.$$

Вариант 9. $x = 0{,}29$; $a = -2{,}4$; $k = 3$; $c = 1{,}52$.

$$y = \sqrt[3]{\frac{\ln x + a^2}{0{,}47x^2}} - \left| 0{,}47x^2 - \frac{10^4}{7} \cos^2 k \right| - \frac{c}{x}.$$

Вариант 10. $a = -2{,}5$; $b = 1{,}35$; $x = 2{,}75$; $i = 3$; $c = -0{,}72$.

$$y = \frac{1{,}5(a-b)^2}{|a-b|c} + \frac{i}{5} + 10\sqrt[3]{|a-b|} + \frac{2{,}5(a+x^2)\sin 7}{ix^2 + a^2 bc}.$$

Вариант 11. $a = 3{,}5$; $i = 2$; $b = -0{,}7$; $x = 0{,}8$.

$$y = 10^4 \sin^2 i - \frac{0{,}32x^3 + 4x + b}{\cos ia} \sqrt[6]{0{,}32x^3 - b + |b|}.$$

Вариант 12. $a = 4{,}72$; $b = 1{,}25$; $d = -0{,}01$; $x = 2{,}25$; $i = 2$; $k = 3$.

$$y = \frac{ax^2 + |d|}{(a+b)^2} - 10^4 \sqrt[5]{\frac{kx}{(a+b)^2}} - \frac{\cos i}{\sin kx}.$$

Вариант 13. $a = -3{,}25$; $x = 8{,}2$; $k = 4$; $b = 0{,}05$; $d = 0{,}95$.

$$y = \cos k(x-a) + \frac{\sqrt[5]{|x+a|}}{2{,}4b} e^3 + 10^{-4} \frac{(x+a)^3 + x^4 d}{k(x-a)^3}.$$

Вариант 14. $x = 0{,}48$; $b = -0{,}31$; $c = 1{,}72$; $a = 2{,}01$; $k = 3$.

$$y = \sqrt[5]{|ax^2 - b^3|} + \ln kx - \frac{e^{kx} + c^2}{\sin kx} - 10^{-3}\sqrt{2157}.$$

Вариант 15. $x = 2{,}5$; $b = 0{,}04$; $k = 3$; $n = 5$.

$$y = \frac{1}{9} + \frac{\sqrt{x^2 + b}}{0{,}4x} - 10^4 e^{kx} + \cos\sqrt{x^2 + b} + \frac{\sin 3}{(x^2 + b)n}.$$

Вариант 16. $x = 0{,}5$; $a = 2{,}71$; $c = 3{,}25$; $d = -3{,}53$; $k = 5$.

$$y = \frac{\sin(ax^2 - c)}{0{,}25k^2 xd} - \left| \sqrt[3]{x^2 + \ln 3} - \cos kx \right| + 10^4 x^5 cd.$$

2.3. Таблицы

Тип данных «таблицы» представляет собой массив в табличной форме, именованные столбцы которой могут иметь различные типы. Это удобный контейнер для хранения данных смешанного типа. Им пользуются и тогда, когда необходимо импортировать в MATLAB данные, хранящиеся во внешнем текстовом файле или электронной таблице.

ВАЖНО! Каждая переменная в таблице может иметь отличный от другой переменной тип данных, в том числе представлять собой матрицу. Обязательное условие ее размещения в таблице – одинаковое число строк с другими переменными таблицы.

2.3.1. Создание таблиц. Функция table

Для создания таблиц используется функция **table**. Она создает табличный массив с именованными переменными, которые могут содержать данные различных типов.

Синтаксис следующий:

```
T = table (var1, ..., varN)
T = table ('Size', sz,' VariableTypes', varTypes)
T = table (___, Name, Value)
T = table
```

Команда **T = table (var1, ..., varN)** составляет таблицу от входных переменных var1, ..., varN. Если входные параметры – это переменные рабочей области, то функция присваивает их имена столбцам в выходной таблице. В противном случае имена столбцов остаются в виде Var1, ..., VarN, где N – количество переменных.

Пример. Из переменных рабочей области создадим и просмотрим таблицу размером 3 × 3, содержащую имя (LastName), фамилию (FirstName) и возраст (Age) трех человек:

```
>> FirstName={'Иванов'; 'Петров'; 'Сидоров'};
>> LastName={'Сергей'; 'Иван'; 'Виктор'};
>> Age=[22; 20; 30];
>> T=table (FirstName, LastName, Age)
T =
  3×3 table
    FirstName      LastName     Age
    _____________    _____________    ___
```

{'Иванов' } {'Сергей'} 22
{'Петров' } {'Иван' } 20
{'Сидоров'} {'Виктор'} 30

Пример. Создадим и просмотрим таблицу с размерами 3 × 2, содержащую значения двух переменных value и s1:

```
value=[1; 2; 3];
s1=[3; 4; 5];
T=table (value, s1)
T =
  3×2 table
    value    s1
    _____    __
      1      3
      2      4
      3      5
```

Альтернативный пример. Преобразуем в таблицу MATLAB данные об успеваемости студентов, содержащиеся в файле «Успеваемость.xlsx». Для этого предварительно разместим файл «Успеваемость.xlsx» в каталоге MATLAB. Содержимое файла представлено на рис. 2.1.

	A	B	C	D	E	F
1	ФИО	21.сен	28.09.2020 зад1	2.10 зад2	02.окт	05.окт
2	Андрианов		2	н		
3	Анисимов		н	н		н
4	Булавинцев		н	н		н
5	Гаврилов		2	4	4	
6	Гулин	4	3	4	5	
7	Дуньямалиев		н	н		н
8	Жуков	н	н	н		н
9	Забелин		3	5	5	
10	Иванов		2	5	4	
11	Исмаилов	н	н	н		н
12	Калабкин		2	3	5	
13	Покровский		2	н		н
14	Польников	н	н	н		н
15	Потин		2	3	5	
16	Цветова	н	н	5		н
17	Мартьянов		н	2		н
18	Тарасов		н	3	5	

Рис. 2.1. Содержимое файла «Успеваемость.xlsx»

Затем наберем команду **T = readtable ('Успеваемость.xlsx')**. Результат работы команды представлен на рис. 2.2.

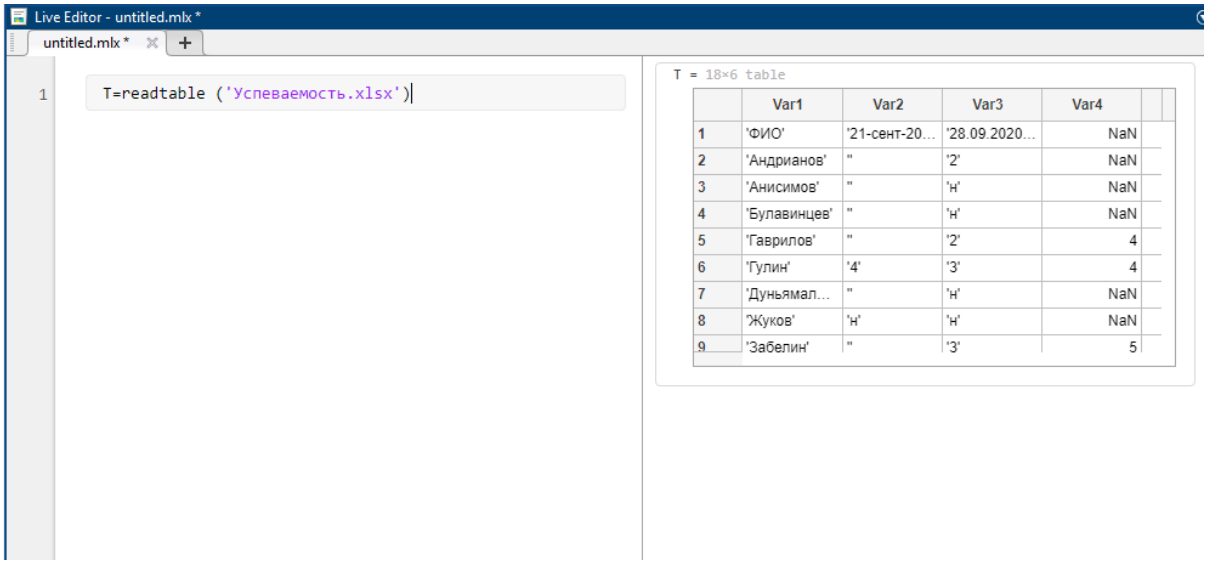


Рис. 2.2. Результат работы функции **readtable**

Как видно по рисунку, имена столбцов исходного файла не являются именами столбцов новой таблицы.

Сохраним полученную таблицу в формате MATLAB, выбрав в панели инструментов вкладку *Save Workspace*. Зададим имя файла – «Успеваемость.mat».

Команда **T = table ('Size', sz, 'VariableTypes', varTypes)** составляет таблицу и предварительно выделяет место для переменных, которые имеют разные типы данных, задаваемые пользователем. sz – двухэлементный числовой массив, где параметр sz(1) задает количество строк, параметр sz(2) – количество переменных. Параметр varTypes задает типы данных переменных.

В качестве примера создадим таблицу размером 3 × 3, содержащую значения температуры за три дня июня 2021 года в городе Владимире:

```
sz = [3 3];
varTypes = {'double','datetime','string'};
varNames = {'Temperature','Time','Station'};
T = table('Size',sz,'VariableTypes',varTypes,'VariableNames',varNames);
```

```
display (T);
T =
  3×3 table
    Temperature    Time     Station
    ___________    ____    _________
        0          NaT     <missing>
        0          NaT     <missing>
        0          NaT     <missing>
```

Заполним столбцы значениями температуры, используя термины «вчера», «сегодня», «завтра»:

```
T(1,:) = {21,datetime('yesterday'),"Владимир"};
T(2,:) = {22,datetime('today'),"Владимир"};
T(3,:) = {28,datetime('tomorrow'),"Владимир"};
display (T);
T =
  3×3 table
    Temperature      Time          Station
    ___________    ___________    __________
        21         14-Jun-2021    "Владимир"
        22         15-Jun-2021    "Владимир"
        28         16-Jun-2021    "Владимир"
```

Более подробно массивы типа datetime рассмотрены ниже (п. 2.4).

Команда **T = table (___, Name, Value)** задает дополнительные входные параметры с помощью одного или нескольких аргументов пары «имя – значение». Можно использовать этот синтаксис с любым из входных параметров предыдущих синтаксисов.

Команда **T = table** составляет пустую таблицу 0 × 0.

Пример. Преобразуем таблицу *T* в массив структур *S*, где каждая именованная переменная становится полем *S*.

```
S = table2struct(T)
S =
  3×1 struct array with fields:
    FirstName
    LastName
    Age
```

2.3.2. Основные функции для работы с таблицами

Основные функции для работы с типом данных «таблица» представлены ниже.

table	Создание табличного массива с именованными переменными, которые могут содержать различные типы данных
array2table	Преобразование гомогенного массива (т. е. массива, в разные элементы которого могут быть непосредственно записаны значения одного типа данных) в таблицу
cell2table	Преобразование массива ячеек в таблицу
struct2table	Преобразование массива структур в таблицу
table2array	Преобразование таблицы в гомогенный массив
table2cell	Преобразование таблицы в массив ячеек
table2struct	Преобразование таблицы в массив структур, где каждая переменная типа «таблица» становится полем структуры
table2timetable	Преобразование таблицы в расписание
timetable2table	Преобразование расписания в таблицу

Вопросы для самоконтроля

1. Чем таблицы отличаются от базовых типов данных MATLAB?
2. Как создать таблицу?
3. Как преобразовать данные базовых числовых типов в таблицу? Когда необходимы такие преобразования? Приведите примеры.

Практическая работа

ТАБЛИЧНЫЙ ТИП ДАННЫХ

Задание. Напишите программу в виде m-файла. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

- текст задания с указанием варианта;
- скрипт, соответствующий содержанию задания;
- результат работы программы.

Варианты заданий

Вариант 1. Создайте таблицу, представляющую сведения о поле (мужской/женский), росте и весе обучаемых. Некоторые данные поля «Пол» могут содержать символы «‘’». Исключите эти символы из данных.

Пол	Рост	Вес

Вариант 2. Представьте сведения в виде двух таблиц (таблица 1 и таблица 2). Объедините таблицы в одну, поместив значения повторяющихся полей в одно поле таблицы 3.

Таблица 1

Фамилия	Пол	Рост	Вес

Таблица 2

Фамилия	Имя	Пол

Таблица 3

Фамилия	Имя	Пол	Рост	Вес

Вариант 3. Создайте таблицу, представляющую сведения о поле (мужской/женский), росте и весе обучаемых. Некоторые данные поля «Рост» содержат символы «‘,’». Организуйте ввод данных согласно таблице и условию задания. Замените символы «‘,’» на символы «‘.’».

Фамилия	Пол	Рост	Вес

Вариант 4. Даны вещественные числа a, $b \neq 0$ и таблица T с размерами $n \times 2$. Организуйте ввод числовых данных по этой таблице. Найдите элемент первого столбца таблицы $T[i, 1]$, равный a. Присвойте переменной b значение элемента $T[i, 2]$.

Вариант 5. Дана таблица T с размерами $n \times 2$. Первый столбец таблицы содержит числовые данные целого типа, второй столбец – числовые данные в виде строки. Приведите таблицу к единообразному виду. Для каждого столбца выясните, имеются ли в нем элементы, большие некоторого числа d.

Вариант 6. Дана таблица размеров женской одежды. Организуйте ввод данных согласно таблице. Напишите программу, выводящую на экран определение размера по введенному значению размера. Например, при вводе чисел 54, 56 программа должна вывести на экран результат – XL.

Определение размера	S		M		L		XL
Российский размер	42 – 44	44 – 46	46 – 48	48 – 50	50 – 52	52 – 54	54 – 56
Английский размер	8	10	12	14	16	18	20
Европейский размер	36	38	40	42	44	46	48

Вариант 7. Дана таблица, содержащая выборочные сведения о 10 пациентах городской больницы: возраст, пол, рост, фамилия, место жительства, оценка здоровья (частота пульса, артериальное давление (два значения – систолическое и диастолическое), температура тела, содержание эритроцитов, лейкоцитов, гормонов), курильщик (да/нет), вес. Организуйте ввод данных согласно таблице. Напишите программу, выводящую на экран список пациентов с удовлетворительной оценкой здоровья.

Вариант 8. Дана таблица, содержащая выборочные сведения о 10 пациентах городской больницы: возраст, пол, рост, фамилия, место жительства, оценка здоровья (частота пульса, артериальное давление (два значения – систолическое и диастолическое), температура тела, содержание эритроцитов, лейкоцитов, гормонов), курильщик (да/нет), вес. Организуйте ввод данных согласно таблице. Напишите программу, выводящую на экран список пациентов женского пола с избыточной массой тела.

Примечание: идеальный вес рассчитывается по индексу массы тела (ИМТ). Необходимо вес, указанный в килограммах, разделить на

рост в квадрате, указанный в метрах. Например, при росте 178 см и весе 69 кг расчет будет таким: ИМТ = 69/(1,78 · 1,78) = 21,78. Индекс массы тела для женщин считается нормальным, если показатель входит в диапазон от 20 до 22.

Вариант 9. Дана таблица, содержащая выборочные сведения о 10 пациентах городской больницы: возраст, пол, рост, фамилия, место жительства, оценка здоровья (частота пульса, артериальное давление (два значения – систолическое и диастолическое), температура тела, содержание эритроцитов, лейкоцитов, гормонов), курильщик (да/нет), вес. Организуйте ввод данных согласно таблице. Напишите программу, выводящую на экран список пациентов конкретного региона, являющихся курильщиками.

Вариант 10. Дана таблица, содержащая выборочные сведения о 10 пациентах городской больницы: возраст, пол, рост, фамилия, место жительства, оценка здоровья (частота пульса, артериальное давление (два значения – систолическое и диастолическое), температура тела, содержание эритроцитов, лейкоцитов, гормонов), курильщик (да/нет), вес. Организуйте ввод данных согласно таблице. Напишите программу, выводящую на экран список пациентов, имеющих риск сердечно-сосудистых заболеваний.

Примечание: систолическим артериальным давлением (САД) называется максимальное давление в артериях в момент, когда сердце сжимается и выталкивает кровь в артерии. Диастолическое артериальное давление (ДАД) показывает давление в артериях в момент расслабления сердечной мышцы, оно отражает сопротивление периферических сосудов. Риск сердечно-сосудистых заболеваний возникает при 130 мм рт. ст. ≤ САД ≤ 139 мм рт. ст. и 85 мм рт. ст. ≤ ДАД ≤ ≤ 89 мм рт. ст.

Вариант 11. Дана таблица, содержащая сведения о студентах вашей группы: фамилия, имя, отчество, название группы, домашний адрес, наличие домашнего телефона (да/нет), домашний телефон. Организуйте ввод данных согласно таблице. В результате работы программы на экран должны быть выведены списки студентов, не имеющих домашнего телефона и имеющих его.

Вариант 12. Напишите программу, которая позволяет хранить данные об автомобилях предприятия (марка, цвет, пробег, год выпуска, водитель). Организуйте ввод данных согласно таблице. Выведите сведения по следующим запросам:

– по фамилии, имени и отчеству (ФИО) водителя – марки машин, которыми он управляет;

– по марке машины – ФИО водителя;

– по году выпуска – марки машин предприятия.

Вариант 13. Составьте программу назначения стипендии студентам по результатам сессии, используя следующие правила:

– если все оценки «5», назначается повышенная стипендия;

– если все оценки «4» и «5», назначается обычная стипендия;

– если есть оценка «3», стипендия не назначается.

В результате работы программы должен быть напечатан список группы с оценками и средними баллами каждого студента и два списка фамилий (назначенных на повышенную и обычную стипендии).

Вариант 14. На аптечном складе хранятся лекарства. Сведения о лекарствах содержатся в специальной таблице: наименование лекарственного препарата, количество, цена, срок хранения (в месяцах). Организуйте ввод данных согласно таблице. Выясните:

– сколько стоят самый дорогой и самый дешевый препараты;

– сколько препаратов хранится на складе;

– какие препараты имеют срок хранения более трех месяцев.

Вариант 15. Торговый склад производит уценку хранящейся продукции. Если продукция хранится на складе дольше n месяцев, то она уценивается в два раза, а если срок хранения превысил m месяцев ($m < n$), но не достиг n, то в 1,5 раза. Организуйте ввод данных согласно ведомости: наименование товара, количество товара, цена товара до уценки, срок хранения товара. Ведомость уценки товаров должна содержать следующую информацию: наименование товара, количество товара, цену товара до уценки, срок хранения товара, цену товара после уценки, общую стоимость товара до уценки, общую стоимость товара после уценки.

Вариант 16. В библиотеке имеются книги. Для каждой книги указаны название, год выпуска, автор(ы). Организуйте ввод данных в таблицу. Выведите информацию о книгах, имеющих одного автора, двух авторов, более трех авторов; изданных позднее 1987 г.

2.4. Представление даты и времени

Массивы значений даты и времени (datetime-массивы) поддерживают разного рода операции и форматы отображения дат и времени. Ниже представлены наиболее часто используемые для этой цели функции.

datetime	Массивы, которые представляют моменты времени
NaT	Не является временем
years	Длительность в годах
days	Длительность в днях
hours	Длительность в часах
minutes	Длительность в минутах
seconds	Длительность в секундах
duration	Отрезки времени в модулях фиксированной длины
calyears	Календарная длительность в годах
calquarters	Календарная длительность в четвертях
calmonths	Календарная длительность в месяцах
calweeks	Календарная длительность в неделях
caldays	Календарная длительность в днях
calendarDuration	Отрезки времени в календарных модулях переменной длины
year	Номер года
quarter	Номер четверти
month	Номер или название месяца
week	Номер недели
day	Номер или название дня
hour	Номер часа
minute	Номер минуты
second	Номер секунды
ymd	Год, месяц и дневные количества datetime

Работа с datetime-массивами аналогична работе с числовыми массивами, т. е. значения даты и времени можно добавить, вычесть, отсортировать, сравнить, конкатенировать. Помимо стандартных функций, для работы с данными этого типа существуют специальные функции, отражающие особенности этого типа данных, например возвращающие информацию о текущей дате, вычисляющие прошлые и будущие даты, преобразующие даты из одного допустимого формата в другой и т. д.

Система MATLAB поддерживает три различных формата представления даты: строковый формат – 'дд-ммм-гггг'; внутренний числовой формат (дата представляется в виде порядкового числа – номера текущего дня, который отсчитывается от 1 января 0000 года); векторный формат – [год месяц день час минута секунда].

MATLAB оперирует с датами, как с порядковыми числами. При этом время рассматривается как дробная часть от порядкового номера дня. Например, порядковый номер 73 1871 соответствует дате 17 октября 2003 года, номер 37 745,75 указывает на момент времени, соответствующий 18 часам 17 октября 2003 года. Продемонстрируем сказанное на примерах.

Пример. Создадим два массива дат и найдем разницу между датами с помощью функции **duration**. Результат сохраним в массиве типа duration (dt):

```
>> t1 = datetime (2007:2010,10,1);
>> t2 = datetime (2014,05,1);
>> dt = t2 - t1
dt =
  1×4 duration array
   57696:00:00   48912:00:00   40152:00:00   31392:00:00
```

Преобразуем полученные даты в года с помощью функции **year ()**:

```
>> Y=years (dt)
Y =
   6.5819    5.5799    4.5805    3.5812
```

Пример. Выделим полное количество лет, применив функцию **fix ()**:

```
>> Y=fix (years (dt))
Y =
    6     5     4     3
```

В MATLAB также предусмотрена возможность представления даты и времени как числовых массивов или текста.

Пример. Преобразуем полученный в примере выше результат в символьный массив, используя явное преобразование типа:

```
Y=char (fix (years (dt)))
Y =
   '_____'
```

Пример. Получим информацию о текущем дне:

```
t1=datetime ('now')
t1 =
  datetime
   05-Oct-2020 16:15:06
```

Пример. Отобразим даты, отстоящие от текущей на +1, +2 и +3 часа. Для этого воспользуемся функцией **hours ()**.

```
t1=datetime ('now')
t1 =
  datetime
05-Oct-2020 16:15:06
>> t2 = t1 + hours (1:3)
t2 =
  1×3 datetime array
   05-Oct-2020 17:15:06   05-Oct-2020 18:15:06   05-Oct-2020 19:15:06
```

Для того чтобы отобразить даты, отстоящие от текущей на –1, –2, –3 часа, воспользуемся функцией **hours ()**. Например:

```
t1=datetime ('now')
t1 =
  datetime
   06-Oct-2020 12:42:24
t2 = t1 - hours (1:3)
t2 =
  1×3 datetime array
   06-Oct-2020 11:42:24   06-Oct-2020 10:42:24   06-Oct-2020 09:42:24
```

Более подробную информацию о рассмотренных и других специальных функциях для работы с датами и временем можно получить, воспользовавшись справочной системой MATLAB или документацией на сайте exponenta.ru.

Вопросы для самоконтроля

1. Когда необходим datetime-массив?
2. Как создать datetime-массив?
3. Чем datetime-массив отличается от данных типа «таблица»?
4. Как добавить значения даты и времени? Приведите пример.

5. Как вычесть или сложить значения даты и времени? Приведите пример.

6. Как сравнить друг с другом значения даты и времени? Приведите пример.

7. Как конкатенировать значения даты и времени? Приведите пример.

8. Какие форматы представления даты поддерживает система MATLAB? Чем они отличаются друг от друга? В каких случаях какой формат лучше?

9. Перечислите и кратко охарактеризуйте основные функции для работы с датами и временем.

Практическая работа

РАБОТА СО ВРЕМЕНЕМ

Задание. Напишите программу в виде m-файла. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий содержанию задания;

○ результат работы программы.

Варианты заданий

Вариант 1. Известно расписание поездов, проходящих через станцию Владимир: номер поезда, место выезда и прибытия (например, «Владимир – Москва»), часы и минуты отправления. Значения часов и минут – целые величины, число часов не превышает 23, число минут – 59. Поезда приходят каждый день. Используя тип данных «datetime-массив», по данному времени определите, какие поезда (номер, назначение) стоят в этот момент в расписании.

Вариант 2. В таблице хранятся результаты переписи населения: фамилия, имя, пол (ж/м), дата рождения. Организуйте ввод данных согласно таблице. Используя тип данных «datetime-массив», проанализируйте данные, выведите статистику по каждому году в период

с 1 января 2000 г. по 1 января 2020 г.: кто рождался чаще – мальчики или девочки? Какой процент от общего количества жителей составили мальчики (девочки)?

Вариант 3. Торговый склад производит уценку хранящейся продукции. Если продукция хранится на складе дольше n месяцев, то она уценивается в два раза, а если срок хранения превысил m месяцев ($m < n$), но не достиг n, то в 1,5 раза. Организуйте ввод данных согласно ведомости: наименование товара, количество товара, цена товара до уценки, дата поступления товара на склад и текущая дата в формате «дд: мм: гг». Ведомость уценки товаров должна содержать следующую информацию: наименование товара, количество товара, цену товара до уценки, срок хранения товара, цену товара после уценки, общую стоимость товара до уценки, общую стоимость товара после уценки.

Вариант 4. В таблице хранятся результаты переписи населения: фамилия, имя, пол (ж/м), дата рождения. Организуйте ввод данных согласно таблице. Проанализируйте данные: кого родилось больше – мальчиков или девочек – в период с 1 января 2000 г. по 1 января 2020 г. Подсчитайте и выведите на экран общее количество жителей, родившихся в заданный период.

Вариант 5. В библиотеке имеются газеты и журналы. Для каждого печатного издания указаны название, дата подписания в печать в формате «дд: мм: гг», ФИО редактора (для газеты), состав редколлегии (для журнала). Организуйте ввод данных согласно таблице. Выведите информацию об изданиях, подписанных в печать в феврале 2000 г.

Вариант 6. В таблице хранятся результаты переписи населения: фамилия, имя, пол (ж/м), дата рождения. Организуйте ввод данных согласно таблице. Проанализируйте данные: какие имена наиболее часто давали мальчикам и девочкам в 1990-е гг.

Вариант 7. Расписание преподавателей представлено в виде таблицы, содержащей день недели, количество пар в этот день, время начала и конца пары в этот день, название дисциплины, ФИО преподавателя. Организуйте ввод данных согласно таблице. Проанализируйте данные: какие дисциплины проводятся более двух раз в неделю? По каким дисциплинам есть практические занятия?

Вариант 8. Есть несколько стандартов измерения и записи времени. В настоящее время применяется стандарт, определяемый атомными часами. Это UTC – «coordinated universal time» – всемирное координированное время.

Даны две таблицы. Первая таблица содержит данные о друзьях: имя, город проживания. Вторая – название города проживания друга, разницу со временем UTC. Например:

'Серёга': 'Омск', 'Соня': 'Москва', 'Дима': 'Челябинск', 'Алина': 'Красноярск', 'Егор': 'Пермь'.

'Санкт-Петербург': 3, 'Москва': 3, 'Самара': 4, 'Новосибирск': 7, 'Екатеринбург': 5. Организуйте ввод данных согласно таблицам. Объедините обе таблицы в одну. Напишите программу, которая по имени друга выведет, сколько у него сейчас времени в формате UTC.

Вариант 9. Таблица содержит поле «Дата» – дату текущего времени ее создания. Напишите программу, добавляющую в это поле даты моментов очередного открытия таблицы.

Вариант 10. Расписание преподавателей представлено в виде таблицы, содержащей день недели, количество пар в этот день, время начала и конца пары в этот день, название дисциплины, ФИО преподавателя. Организуйте ввод данных согласно таблице. Проанализируйте данные: по каким дисциплинам есть лабораторные и практические занятия? Какой процент от общего количества дисциплин они составляют?

Вариант 11. С клавиатуры в случайном порядке вводятся даты. Создайте datetime-массив с полями в формате 'yyyy'-'MM'-'dd', 'HH':'mm':'ss', упорядоченными по возрастанию – от начальной до конечной даты.

Вариант 12. Год високосный, если его номер кратен 4. Из кратных 100 високосные лишь те года, которые кратны также 400. Например, 1700, 1800 и 1900 – не високосные года, 2000 – високосный. С клавиатуры в случайном порядке вводятся даты. Создайте datetime-массив с полями в формате 'yyyy'-'MM'-'dd', 'HH':'mm':'ss', содержащий только високосные года. Проанализируйте данные: какой процент из них кратен 100 и не кратен 4?

Вариант 13. Работа светофора для пешеходов запрограммирована следующим образом: в начале каждого часа в течение трех минут горит зеленый свет, затем в течение двух минут – красный, в те-

чение одной минуты – желтый, а затем опять зеленый и т. д. Дано вещественное число *t*, означающее время в минутах, прошедшее с начала очередного часа. Создайте datetime-массив, значения полей которого соответствуют работе светофора. Проанализируйте данные: сигнал какого цвета горел чаще остальных в указанное время *t*?

Вариант 14. Расписание преподавателей представлено в виде таблицы, содержащей день недели, количество пар в этот день, время начала и конца пары в этот день, название дисциплины, ФИО преподавателя. Организуйте ввод данных согласно таблице. Проанализируйте данные: какие дисциплины и сколько раз проводятся в течение недели с 10:20 до 11:50?

Вариант 15. Расписание преподавателей представлено в виде таблицы, содержащей дату, количество пар в этот день, время начала и конца пары в этот день, название дисциплины, ФИО преподавателя. Организуйте ввод данных согласно таблице. Выведите дату и время начала и окончания занятий, суммарное число часов, отработанных преподавателем в этот день. Проанализируйте данные: в какой(ие) день(дни) преподаватель отработал наибольшее количество часов?

Вариант 16. Известно расписание поездов, проходящих через станцию Владимир: номер поезда, место выезда и прибытия, (например, «Владимир – Москва»), часы и минуты отправления. Значения часов и минут – целые величины, число часов не превышает 23, число минут – 59. Поезда приходят каждый день. Возможны задержки прибытия поезда. Используя тип данных «datetime-массив», по данному времени определите, какие поезда (номер, назначение) задерживаются в данный момент времени.

2.5. Категориальные массивы

Массивы типа categorical используются для хранения категорий данных, значения которых – дискретные величины. Ниже представлены основные функции для работы с категориальными массивами.

categorical	Создает массив, который содержит значения, присвоенные категориям
iscategorical	Определяет, является ли введенный массив категориальным массивом
discretize	Группирует данные в интервалы или категории
categories	Категории категориального массива

isordinal	Определяет, является ли введенный массив порядковым категориальным массивом
addcats	Добавляет категории в категориальный массив
mergecats	Объединяет категории в категориальном массиве
removecats	Удаляет категории из категориального массива
renamecats	Переименовывает категории в категориальном массиве
reordercats	Переупорядочивает категории в категориальном массиве
setcats	Устанавливает категории в категориальном массиве
isundefined	Находит неопределенные элементы в категориальном массиве
summary	Выводит сводные данные таблицы, расписания или категориального массива

Рассмотрим некоторые функции.

Создание категориального массива. Синтаксис следующий:

```
B = categorical (A)
B = categorical (A, valueset)
B = categorical (A, valueset, catnames)
B = categorical (A ,___, Name, Value)
```

Команда **B = categorical (A)** создает категориальный массив из массива *A*. Массив *B* содержит только уникальные значения массива *A*.

Пример. Создадим категориальный массив любимых пород кошек:

```
cat={'персидская', 'бенгальская', 'сфинкс'};
cat=categorical (cat);
display (cat);
cat =
  1×3 categorical array
    бенгальская     персидская     сфинкс
```

Категории не имеют никакого математического упорядоченного расположения. MATLAB располагает категории в алфавитном порядке, т. е. на выходе получим результат в следующем виде: {'бенгальская' 'персидская' 'сфинкс'}. Для перечисления категорий используется функция **categories (cat)**:

```
categories(cat)
ans =
  3×1 cell array
```

```
    {'бенгальская'}
    {'персидская' }
    {'сфинкс'}
```

Команда **B = categorical (A, valueset)** создает категориальный массив из массива *A*, где valueset – название категории. Категории массива *B* находятся в том же порядке, что и значения valueset. Например:

```
%дан А-массив ячеек из символьных векторов
A = {'republican' 'democrat'; 'democrat' 'democrat'; 'democrat' 'republican'};
%задаем категории в массиве B
valueset = {'democrat' 'republican' 'independent'};
%преобразуем массив А к категориальному
B = categorical(A,valueset);
%отображаем категории
categories(B)
3×1 cell array
    {'democrat'   }
    {'republican' }
    {'independent'}
```

Можно использовать valueset для того, чтобы включать в массив *B* категории для значений, не существующих в массиве *A*:

```
 %дан А-массив ячеек из символьных векторов
A = {'republican' 'democrat'; 'democrat' 'democrat'; 'democrat' 'republican'};
%добавим в массив B категорию undetectable, отличную от значений A
valueset = {'democrat' 'republican' 'independent' 'undetectable'};
%преобразуем массив А к категориальному
B = categorical(A,valueset);
%отображаем категории
categories(B)
ans =
  4×1 cell array
    {'democrat'    }
    {'republican'  }
    {'independent' }
    {'undetectable'}
```

Если массив *A* содержит значения, не существующие в valueset, соответствующие элементы массива *B* не будут определены:

```
>> categ_array
Error using vertcat
Dimensions of arrays being concatenated are not consistent.
Error in categ_array (line 2)
A = {'democrat'; 'democrat' 'democrat'; 'democrat' };
```

Команда **B = categorical (A, valueset, catnames)** называет категории valueset в соответствии с именами в категории catnames. В примере ниже значение «1» массива *A* будет являться категорией в массиве *B* и называться red; значение «2» – green, а значение «3» – blue:

```
%A-массив чисел
A = [1 3 2;  2 1 3;  3 1 2];
B = categorical(A, [1 2 3], {'red' 'green' 'blue'});
categories (B)
3×1 cell array
   {'red'  }
   {'green'}
   {'blue' }
```

Команда **B = categorical (A, ___ , Name, Value)** создает категориальный массив с дополнительными опциями, заданными одним или несколькими парными аргументами Name,Value. Можно включать любой из входных параметров из предыдущих синтаксисов. Например, создадим числовой массив *A* [5; 2]. Преобразуем массив *A* к порядковому категориальному массиву, где 1, 2 и 3 представляют собой категории child, adult и senior соответственно:

```
A = [3 2;3 3;3 2;2 1;3 2];
display (A);
valueset = [1:3];
catnames = {'child' 'adult' 'senior'};
B = categorical(A,valueset,catnames);
B
A =
    3     2
    3     3
    3     2
    2     1
    3     2
B =
  5×2 categorical array
    senior      adult
```

```
    senior      senior
    senior      adult
    adult       child
    senior      adult
```

Для указания на то, что категории имеют математическое упорядоченное расположение, задается опция 'Ordinal', true. Таким образом, категории массива *B* имеют математическое упорядоченное расположение, т. е. child < adult < senior:

```
A = [3 2;3 3;3 2;2 1;3 2];
valueset =[1:3];
catnames = {'child' 'adult' 'senior'};
B = categorical(A,valueset,catnames,'Ordinal',true)
5×2 categorical array
    senior      adult
    senior      senior
    senior      adult
    adult       child
    senior      adult
```

Создадим массив, содержащий две одинаковые категории «сфинкс». Выведем на экран сводные данные о категориях в исходном массиве cat, используя функцию **summary ()**. Затем удалим лишнюю категорию с помощью функции **removecats ()**:

```
cat={'персидская', 'бенгальская', 'сфинкс', 'сфинкс'};
>> cat=categorical (cat)
cat =
  1×4 categorical array
персидская    бенгальская    сфинкс    сфинкс
summary (cat)
бенгальская    персидская    сфинкс
      1             1          2
cat1=removecats (cat)
categories (cat1)
ans =
  3×1 cell array
    {'бенгальская'}
    {'персидская'}
    {'сфинкс' }
```

Более подробную информацию о рассмотренных и других специальных функциях для работы с категориями можно получить, воспользовавшись справочной системой MATLAB или документацией на сайте exponenta.ru.

Вопросы для самоконтроля

1. Когда необходим категориальный массив?
2. Как создать массив категорий? Приведите пример.
3. Чем категориальный массив отличается от данных типа «datetime-массив», «таблица»?
4. Как добавить (удалить, совместить) категории? Приведите пример.
5. Как сравнить категории друг с другом? Приведите пример.
6. Перечислите и кратко охарактеризуйте основные функции для работы с категориальными массивами.

Практическая работа

РАБОТА С КАТЕГОРИАЛЬНЫМИ МАССИВАМИ

Задание. Напишите программу в виде m-файла. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий содержанию задания;

○ результат работы программы.

Варианты заданий

Вариант 1. Жанры кино представлены в таблице.

№	Жанр кино
1	Биографический
2	Исторический
3	Военный
4	Детектив
5	Документальный
6	Драма

Преобразуйте таблицу в категориальный массив. Напишите программу, организующую ввод данных согласно новой таблице, например, как показано ниже.

Биографический	Исторический	Военный	Детектив	Документальный	Драма
Сид и Нэнси	Щитом и мечом	Щит и меч	Пуаро	Путешествия на край Вселенной	Волк с Уолл-стрит
Список Шиндлера	София	…	Мисс Марпл Агаты Кристи	Океаны	…
…	…	…	…	Сказания о динозаврах	…
…	…	…	…	…	…

Проанализируйте данные: фильмов какого жанра(ов) в таблице больше (одинаковое количество, меньше)?

Вариант 2. На вход подаются данные: фамилия, имя, отчество, пол, социальный статус. Организуйте распределение вводимых данных по двум категориям: married (замужем/женат) и single (одинока/одинок). Упорядочьте введенные данные по алфавиту. Проанализируйте данные: кого больше – женатых мужчин или разведенных женщин?

Вариант 3. Категории отходов представлены в таблице ниже.

Преобразуйте таблицу в категориальный массив. Добавьте в массив дополнительные категории: стекло, органические отходы, кожа, опасные бытовые отходы. Организуйте ввод данных согласно категориям. Проанализируйте данные: какой процент от общего количества составляет каждая категория?

№	Вид отходов
1	Бумага
2	Пластик
3	Металл

Вариант 4. Дан массив действительных чисел

A = [–3 2 3 0 2,2 –13 2,87 4,657].

Создайте категориальный массив B с категориями: «положительные числа», «отрицательные числа», «ноль». Распределите значения элементов массива A по категориям массива B. В каждой категории упорядочьте данные по убыванию их значений.

Вариант 5. Создайте категориальный массив B с категориями: «положительные числа», «отрицательные числа», «ноль». Организуйте ввод данных согласно категориям. Продублируйте все введенные данные с сохранением порядка их следования.

Вариант 6. Билет с шестизначным цифровым номером считается «счастливым», если сумма трех старших цифр равна сумме трех младших цифр. Создайте категориальный массив *B* с категориями «счастливые» и «несчастливые» билеты. Организуйте ввод и распределение шестизначных номеров по указанным категориям. Определите макимальный и минимальный номера в каждой категории.

Вариант 7. Создайте категориальный массив *B* с категориями: «положительные числа» и «отрицательные числа». Организуйте ввод данных согласно категориям. Проанализируйте введенные данные: сколько в каждой категории одинаковых соседних элементов?

Вариант 8. Дана матрица *A* с размерами 5 × 5, состоящая из случайных чисел. Составьте категориальный массив, в котором будут содержаться значения суммы элементов каждого столбца матрицы *A*. Упорядочьте категории массива по возрастанию значения суммы.

Вариант 9. Имеется два массива целых чисел. Создайте из них категориальный массив с категориями: «положительные числа», «отрицательные числа», «ноль». Организуйте ввод данных согласно категориям. Для каждой категории выведите на экран данные, наиболее отклоняющиеся от максимального значения.

Вариант 10. Создайте категориальный массив *B* с категориями: «положительные числа» и «отрицательные числа». Организуйте ввод данных согласно категориям. Вставьте на *k*-е места элементы, равные наименьшему элементу в каждой категории.

Вариант 11. Имеется два массива действительных чисел. Создайте из них категориальный массив с категориями: «положительные числа», «отрицательные числа», «ноль». Организуйте ввод данных согласно категориям. Объедините категории «положительные числа» и «ноль» в категорию «положительные числа». Удалите из этой категории все дробные числа.

Вариант 12. Создайте категориальный массив *C* с категориями: «положительные числа» и «отрицательные числа». Организуйте ввод данных согласно категориям. Проанализируйте данные: в каждой категории определите, какой из элементов повторяется наибольшее количество раз. Удалите его из категории.

Вариант 13. Создайте категориальный массив *B* с категориями: «положительные числа» и «отрицательные числа». Организуйте ввод

данных согласно категориям. В каждой категории замените все элементы, меньшие среднего арифметического, округленного до целого.

Вариант 14. В категориальном массиве хранятся сведения о количестве осадков, выпавших за первую, вторую и третью декады месяца. Организуйте ввод данных согласно категориям. Определите: общее количество осадков за месяц (декаду); среднедекадное количество осадков; категорию с наибольшим количеством осадков.

Вариант 15. В трех массивах хранятся сведения о 12 различных предметах: название, форма, цвет, размер. Сведения могут повторяться. Создайте категориальный массив *C* с категориями: «форма» и «размер». Организуйте ввод данных, содержащихся в массивах, согласно категориям. Проанализируйте данные: какой предмет встречается чаще (меньше) всего?

Вариант 16. В категориальном массиве хранится информация о книгах в каждом из пяти разделов библиотеки. Организуйте ввод данных согласно категориям. В каждой категории упорядочьте сведения в алфавитном порядке по фамилии первого автора. Проанализируйте данные: книги какого автора наиболее поздние?

2.6. Расписания

Расписания – это данные, представленные в табличной форме и содержащие метку времени. Примером данных этого типа могут служить данные о погодных условиях, записанные в разное время суток.

Так же как и таблицы, расписания хранят данные смешанного типа. Помимо этого, они обеспечивают специфичные для времени функции, позволяющие выровнять, объединить данные с меткой времени и выполнить вычисления с ними в одном или нескольких расписаниях.

Ниже приведем описание некоторых функций, используемых для работы с этим типом данных.

timetable	Массив расписания со строками с меткой времени и переменными различных типов
table2timetable	Преобразование таблицы в расписание
array2timetable	Преобразование гомогенного массива в расписание
timetable2table	Преобразование расписания в таблицу
istimetable	Определение, являются ли введенные данные расписанием

summary	Вывод сводных данных таблицы, расписания или категориального массива
retime	Передискретизация или агрегирование данных в расписании
synchronize	Синхронизация расписания с общим временным вектором и передискретизация или агрегирование данных из входных расписаний
lag	Данные сдвига времени в расписании

Рассмотрим те из них, которые наиболее важны для анализа данных.

Пример. Создадим таблицу, содержащую данные о дате и времени измерения (MeasurementTime), температуре (Temp), давлении (Pressure) и скорости ветра (WindSpeed):

```
MeasurementTime = datetime ({'2015-12-18 08:03:05'; '2015-12-18 10:03:17'; '2015-12-18 12:03:13'});
Temp = [37.3; 39.1; 42.3];
 Pressure = [30.1; 30.03; 29.9];
WindSpeed = [13.4; 6.5; 7.3];
TT = timetable (MeasurementTime, Temp, Pressure, WindSpeed)
TT =
  3×3 timetable
    MeasurementTime       Temp    Pressure    WindSpeed
    ____________________    ____    ________    _________

    18-Dec-2015 08:03:05    37.3     30.1        13.4
    18-Dec-2015 10:03:17    39.1     30.03        6.5
    18-Dec-2015 12:03:13    42.3     29.9         7.3
```

Для того чтобы восстановить данные о температуре, давлении и скорости ветра с постоянным временным интервалом, например в один час, воспользуемся специальной фунцией **retime ()**. Например:

```
TT2 = retime (TT,'hourly','linear')
TT2 =
  6×3 timetable
    MeasurementTime       Temp     Pressure    WindSpeed
    ____________________    ______    ________    _________

    18-Dec-2015 08:00:00    37.254     30.102      13.577
```

```
    18-Dec-2015 09:00:00    38.152    30.067    10.133
    18-Dec-2015 10:00:00    39.051    30.032    6.6885
    18-Dec-2015 11:00:00    40.613    29.969    6.8783
    18-Dec-2015 12:00:00    42.214    29.903    7.2785
    18-Dec-2015 13:00:00    43.815    29.838    7.6788
```

Распечатаем с помощью функции **summary** сводную информацию о таблице. Для каждого столбца будут выведены минимальное, среднее и максимальное значения:

```
RowTimes:
    MeasurementTime: 6×1 datetime
        Values:
            Min          18-Dec-2015 08:00:00
            Median       18-Dec-2015 10:30:00
            Max          18-Dec-2015 13:00:00
            TimeStep     01:00:00
Variables:
    Temp: 6×1 double
        Values:
            Min       37.254
            Median    39.832
            Max       43.815
    Pressure: 6×1 double
        Values:
            Min       29.838
            Median       30
            Max       30.102
    WindSpeed: 6×1 double
        Values:
            Min       6.6885
            Median    7.4787
            Max       13.577
```

Более подробную информацию о рассмотренных и других специальных функциях для работы с датой и временем можно получить, воспользовавшись справочной системой MATLAB или документацией на сайте exponenta.ru.

Вопросы для самоконтроля

1. Чем расписания отличаются от таблиц, datetime-массивов (категориальных массивов)?

2. Как создать массив расписаний?

3. Как преобразовать массив расписаний в таблицу, категориальный массив? Когда необходимы такие преобразования? Приведите примеры.

4. Можно ли получить данные о сдвиге времени в расписании? Расскажите, как это можно сделать.

5. Какую важную для анализа данных информацию можно получить с помощью специальных функций? Перечислите их. Приведите примеры использования.

Практическая работа

РАБОТА С ДАННЫМИ ТИПА «РАСПИСАНИЕ»

Задание. Напишите скрипт. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий содержанию задания;

○ результат работы программы.

Варианты заданий

Вариант 1. Известно расписание поездов, проходящих через станцию Владимир: номер поезда, место выезда и прибытия (например, «Владимир – Москва»), часы и минуты отправления. Значения часов и минут – целые величины, число часов не превышает 23, число минут – 59. Поезда приходят каждый день. Используя тип данных «расписание», по данному времени определите, какие поезда (номер, назначение) стоят в этот момент в расписании.

Вариант 2. Торговый склад производит уценку хранящейся продукции. Если продукция хранится на складе дольше n месяцев, то

она уценивается в два раза, а если срок хранения превысил m ($m < n$) месяцев, но не достиг n, то в 1,5 раза. Организуйте ввод данных согласно ведомости: наименование товара, количество товара, цена товара до уценки, дата поступления товара на склад и текущая дата в формате «дд: мм: гг». Ведомость уценки товаров должна содержать следующую информацию: наименование товара, количество товара, цену товара до уценки, срок хранения товара, цену товара после уценки. Используйте тип данных «расписание».

Вариант 3. В библиотеке имеются газеты и журналы. Для каждого печатного издания указаны название, дата подписания в печать в формате «дд: мм: гг», ФИО редактора (для газеты), состав редколлегии (для журнала). Используя тип данных «расписание», выведите информацию об изданиях, подписанных в печать в феврале 2000 г.

Вариант 4. На предприятии семь суток в неделю рабочие работают по графику в три смены: ночная – 00:00 – 08:00, дневная – 08:00 – 16:00, вечерняя – 16:00 – 00:00. Используя тип данных «расписание», минимизируйте общее количество рабочих на предприятии; распределите рабочих так, чтобы каждый день в каждую из смен присутствовало определенное количество работников согласно графику, представленному в таблице.

Смена	Количество работников по дням недели, чел.						
	Понедельник	Вторник	Среда	Четверг	Пятница	Суббота	Воскресенье
Ночная	5	3	2	4	3	2	2
Дневная	7	8	9	5	7	2	5
Вечерняя	9	10	10	10	7	1	2

При распределении необходимо учесть следующие ограничения:

– в течение суток работник может работать только в одной смене;

– расписание фиксированное в течение четырех дней.

Вариант 5. В таблице хранятся результаты переписи населения: фамилия, имя, пол (ж/м), дата рождения. Организуйте ввод данных согласно таблице. Используя тип данных «расписание», проанализируйте данные, выведите статистику по каждому году за период с 1 января 2000 г. по 1 января 2020 г.: кто рождался чаще – мальчики или девочки? Какой процент от общего количества жителей составили мальчики (девочки)?

Вариант 6. Торговый склад производит уценку хранящейся продукции. Если продукция хранится на складе дольше n месяцев, то она уценивается в два раза, а если срок хранения превысил m ($m < n$) месяцев, но не достиг n, то в 1,5 раза. Организуйте ввод данных согласно ведомости: наименование товара, количество товара, цена товара до уценки, дата поступления товара на склад и текущая дата в формате «дд: мм: гг». Ведомость уценки товаров должна содержать следующую информацию: наименование товара, количество товара, цену товара до уценки, срок хранения товара, цену товара после уценки, общую стоимость всех товаров до уценки, общую стоимость всех товаров после уценки. Используйте тип данных «расписание».

Вариант 7. В таблице хранятся результаты переписи населения: фамилия, имя, пол (ж/м), дата рождения. Организуйте ввод данных согласно таблице. Проанализируйте данные: кого родилось больше – мальчиков или девочек – в период с 1 января 2000 г. по 1 января 2020 г. Используя тип данных «расписание», подсчитайте и выведите на экран общее количество жителей, родившихся после 1 января 2001 г.

Вариант 8. В библиотеке имеются газеты и журналы. Для каждого печатного издания указаны название, дата подписания в печать в формате «дд: мм: гг», ФИО редактора (для газеты), состав редколлегии (для журнала). Организуйте ввод данных согласно таблице. Используя тип данных «расписание», выведите информацию об изданиях, подписанных в печать с февраля по март 2000 г.

Вариант 9. В таблице хранятся результаты переписи населения: фамилия, имя, пол (ж/м), дата рождения. Организуйте ввод данных согласно таблице. Используя тип данных «расписание», проанализируйте данные: какие имена наиболее часто давали мальчикам и девочкам в 1990-е г.

Вариант 10. Расписание преподавателей представлено в виде таблицы, содержащей день недели, количество пар в этот день, время начала и конца пары в этот день, название дисциплины, ФИО преподавателя. Организуйте ввод данных согласно таблице. Используя тип данных «расписание», проанализируйте данные: какие дисциплины проводятся более двух раз в неделю? По каким дисциплинам есть практические занятия?

Вариант 11. Есть несколько стандартов измерения и записи времени. В настоящее время применяется стандарт, определяемый

атомными часами. Это UTC – «coordinated universal time» – всемирное координированное время.

Даны две таблицы. Первая таблица содержит данные о друзьях: имя, город проживания. Вторая – название города проживания друга, разницу со временем UTC. Например:

'Серёга': 'Омск', 'Соня': 'Москва', 'Дима': 'Челябинск', 'Алина': 'Красноярск', 'Егор': 'Пермь'.

'Санкт-Петербург': 3, 'Москва': 3, 'Самара': 4, 'Новосибирск': 7, 'Екатеринбург': 5. Организуйте ввод данных согласно таблицам. Объедините обе таблицы в одну. Используя тип данных «расписание», напишите программу, которая по имени друга выведет, сколько у него сейчас времени в формате UTC.

Вариант 12. Таблица содержит поле «Дата», в котором указана дата текущего времени ее создания. Используя тип данных «расписание», напишите программу, добавляющую в это поле даты очередного открытия таблицы.

Вариант 13. Расписание преподавателей представлено в виде таблицы, содержащей день недели, количество пар в этот день, время начала и конца пары в этот день, название дисциплины, ФИО преподавателя. Организуйте ввод данных согласно таблице. Используя тип данных «расписание», проанализируйте данные: по каким дисциплинам есть лабораторные и практические занятия? Какой процент от общего количества дисциплин они составляют?

Вариант 14. Данные о погодных условиях хранятся в переменных рабочей области: data_time (дата, время), temperature (температура воздуха), pressure (давление), windspeed (скорость ветра). Создайте массив-расписание из переменных рабочей области. Синхронизируйте значения температуры воздуха, давления и скорости ветра с соответствующими датами и временем. Используя тип данных «расписание», проанализируйте данные: в какое время суток (дневное, ночное или вечернее) скорость ветра была наибольшей (наименьшей)?

Вариант 15. Создайте ежедневник со списком дел в виде datetime-массива с полями в формате 'yyyy'-'MM'-'dd', 'HH':'mm':'ss' от начальной даты на *n* дней вперед с несколькими значениями 'HH':'mm':'ss' для каждого дня и соответствующими отведенному времени делами.

Организуйте вывод данных за весь период времени, на каждый затребованный пользователем день. Проанализируйте данные: какие дела выполнялись чаще всего за *n* дней?

Вариант 16. Создайте ежедневник со списком дел в виде datetime-массива с полями в формате 'yyyy'-'MM'-'dd', 'HH':'mm':'ss' от начальной даты на *n* дней вперед с несколькими значениями 'HH':'mm':'ss' для каждого дня и соответствующими отведенному времени делами. Организуйте вывод дел на текущий момент времени. Проанализируйте данные: какие текущие дела выполнялись чаще всего за *n* дней?

2.7. Структуры

Структуры – это массивы с именованными полями, которые могут содержать данные любого типа и размера. Переменные, входящие в состав структуры, называются *полями структуры*.

На рисунке 2.3 представлена схема структуры patient (пациент), хранящая данные о пациенте, с полями name (имя), billing (счет), test (анализы).

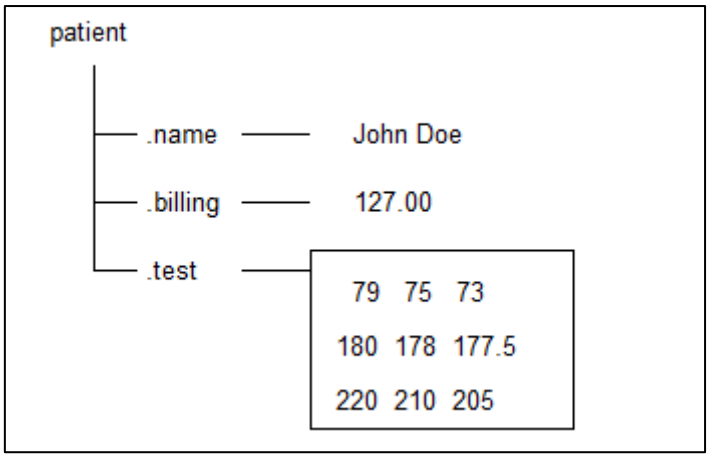


Рис. 2.3. Схема структуры patient

Схема показывает, как структура хранит данные. Так как переменная patient хранит одну структуру, она называется *скалярной структурой*.

Как видно по рис. 2.3, для добавления полей name, billing и test к структуре используется символ «.». Например:

```
patient.name = 'John Doe';
```

patient.billing = 127;
patient.test = [79 75 73; 180 178 177.5; 220 210 205]

После того как поле создано, символ «.» используется для доступа и изменения значения поля. В качестве примера изменим значение поля billing:

patient.billing = 512.00

Вывести значения полей структуры можно с помощью функции **disp ()**:

disp (patient.name)
disp (patient.billing)
disp (patient.test)

Операции с полями и значениями полей выполняются по тем же правилам, что и при работе с обычными массивами. Однако есть ряд функций, осуществляющих специфические для структур операции. Перечень функций и их описание представлены ниже.

struct	Массив структур
fieldnames	Возвращает имена полей структуры или общедоступные поля Java или Microsoft COM object
getfield	Возвращает содержимое полей массива структур
isfield	Определяет, является ли введенное имя именем поля массива структур
isstruct	Определяет, является ли введенный массив массивом структур
orderfields	Упорядочивает поля массива структур
rmfield	Удаляет поля из структуры
setfield	Присваивает значение полю массива структур
arrayfun	Применяет функцию ко всем элементам массива структур
structfun	Применяет функцию к каждому полю скалярной структуры
table2struct	Преобразует таблицу в массив структур
struct2table	Преобразует массив структур в таблицу
cell2struct	Преобразует массив ячеек в массив структур
struct2cell	Преобразует структуру в массив ячеек

Рассмотрим на примерах работу некоторых из перечисленных функций.

Функция **struct ()** используется для создания структур. Задается одним из способов, указанных ниже:

– **имя переменной = struct** создает скалярную структуру с размерами 1 × 1 без полей;

– **struct (field, value)** создает массив структур с одним заданным полем field и соответствующим ему значением value;

– **struct ('field1', value1, 'field2', value2, …, fieldn, valuen)** создает массив структур с несколькими полями;

– **struct ([])** создает пустую структуру с размерами 0 × 0 без полей;

– **struct(OBJ)** конвертирует объект OBJ в эквивалентную структуру или массив структур. OBJ может быть объектом или массивом Java.

Пример. Создадим массив структур без полей. Затем зададим два поля *a* и *b*, одно из которых имеет целый тип, а другое – строковый. Выведем значения полей на экран:

```
s=struct;
s.a=1;
s.b={'A','B'};
disp(s)
a: 1
   b: {'A'  'B'}
s =
  struct with fields:
    a: 1
    b: {'A'  'B'}
```

Создадим массив структур с заранее заданным полем (mycell) и значениями поля, хранящимися в переменной value. Выведем на экран:

```
field = 'mycell';
value = {{'a','b','c'}};
s = struct(field,value);
disp(s);
mycell: {'a'  'b'  'c'}
```

Пример. Пусть дан массив структур из примера выше. Проверим, являются ли имена *c* и *b* именами полей структуры *s*.

```
s=struct;
s.a=1;
s.b={'A','B'};
TF = isfield(s,'c');
TF
TF =
  logical
   0
TF=isfield (s,'b')
TF
TF =
  logical
   1
```

Как видно из примера, функция **isfield ()** возвращает значение «1» (true), если заданное имя – имя поля, и значение «0» (false) – в противном случае.

Пример. Аналогичным образом проверим, является ли заданный массив массивом структур:

```
patient=struct;
patient.name = 'John Doe';
patient.billing = 127.00;
patient.test = [79 75 73; 180 178 177.5; 220 210 205];
isstruct (patient)
ans =
1
```

Пример. Имеется массив структур с одним полем, представляющим собой массив вещественных чисел. Каждый элемент массива заменим на его целое значение, например с помощью одной из функций округления **round ()**, где @round – указатель на функцию **round ()**:

```
s=struct;
s.A=[2.4  3.2  2 5.6];
s.A = arrayfun (@round, s.A);
s
s =
struct with fields:
A: [2 3 6]
```

Пример. Имеется массив структур с двумя полями, представляющими собой массивы вещественных и целых чисел соответственно. Вычислим среднее значение в массиве *A* и средние значения в каждом массиве с помощью функций **structfun ()** и **mean ()**, где @mean – указатель на функцию **mean ()**.

```
s=struct;
s.A=[2.4 3.22 5.6];
s.A=structfun(@mean,s);
s.A
3.7400
s.B=[7 8 9];
s.B = structfun(@mean,s);
s.B
3.7400
8.0000
```

Массивы структур находят самое широкое применение. Например, они используются для представления цветных изображений модели RGB. Они состоят из массивов интенсивности трех цветов – красного r, зеленого g и синего b. Еще более сложные структуры необходимы для разработки баз данных, например о работниках предприятия, службах города, городах страны и т. д. Во всех этих случаях особенно важны возможности доступа к отдельным ячейкам структур и присвоения таким структурам уникальных имен.

Вопросы для самоконтроля

1. Чем структуры отличаются от таблиц, datetime-массивов (категориальных массивов)?

2. Как создать структуру?

3. Как преобразовать массив структур в таблицу, категориальный массив? Когда необходимы такие преобразования? Приведите примеры.

4. Какую важную для анализа данных информацию можно получить с помощью специальных функций? Перечислите их. Приведите примеры использования.

Практическая работа

СТРУКТУРЫ

Задание. Напишите программу. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий содержанию задания;

○ результат работы программы.

Варианты заданий

Вариант 1. Задайте массив структур, содержащий сведения о работниках предприятия. Выведите: список бухгалтеров, работников в возрасте от 30 до 50 лет, среднюю заработную плату по предприятию.

№	ФИО	Должность	Дата рождения	Зарплата

Вариант 2. Багаж пассажира характеризуется количеством и общим весом вещей. Имеется информация о багаже нескольких пассажиров – соответствующие пары чисел. Задайте структуру по условию задачи. Организуйте ввод данных согласно полям структуры. Проанализируйте данные: подсчитайте и сохраните в виде поля структуры общее количество вещей каждого пассажира; выясните, имеется ли пассажир, багаж которого состоит из одной вещи весом не менее 30 кг.

Вариант 3. Задайте массив структур из 25 записей, содержащий следующие сведения: фамилия, имя, адрес и номер телефона. Организуйте ввод данных согласно полям структуры. Проанализируйте данные: найдите и выведите на экран фамилии и адреса людей, чей телефон начинается с цифры 3. Рассмотрите случаи, когда телефон задан: а) в виде семизначного числа; б) с разделяющими цифры дефисами, например 268-50-59.

Вариант 4. Задайте массив структур из 20 записей, содержащий данные о сотрудниках фирмы (фамилия, имя, заработная плата, пол). Организуйте ввод данных согласно полям структуры. Проанализируйте данные: определите фамилию (фамилии) мужчины (мужчин), имеющего (имеющих) самую большую заработную плату; фамилии мужчин и женщин, имеющих самую маленькую заработную плату.

Вариант 5. Задайте массив структур из 16 записей, содержащий данные о сотрудниках фирмы: фамилия, имя, возраст и отношение к воинской службе (военнообязанный или нет). Организуйте ввод данных согласно полям структуры. Проанализируйте данные: определите фамилию и имя самого младшего по возрасту военнообязанного; фамилии самых старших по возрасту людей среди военнообязанных и невоеннообязанных.

Вариант 6. Заданы названия 26 городов и страны, в которых они находятся. Создайте структуру, соответствующую условию задачи. Организуйте ввод данных согласно полям структуры. Преобразуйте данные: распределите города по категориям – названиям стран, где они расположены. Проанализируйте данные: городов какой страны больше (меньше) всего?

Вариант 7. Заданы названия 26 городов и страны, в которых они находятся. Создайте структуру, соответствующую условию задачи. Организуйте ввод данных согласно полям структуры. Преобразуйте данные: распределите города по категориям – «начинается с буквы А», «начинается с буквы В», «начинается с буквы С», «прочие». Проанализируйте данные: каких городов больше (меньше) всего?

Вариант 8. Известны данные о численности населения (миллионы жителей) и площади государства (тысячи квадратных километров). Создайте структуру, соответствующую условию задачи. Организуйте ввод данных согласно полям структуры. Проанализируйте данные: определите плотность населения в каждом государстве; названия государств с наименьшей плотностью населения и их процент от общего количества.

Вариант 9. Задайте массив структур из 30 записей, содержащий данные о сотрудниках фирмы: фамилия, имя, домашний адрес. Организуйте ввод данных согласно полям структуры. Проанализируйте данные: работают ли в фирме люди с фамилиями Кузин, Куравлев,

Кульков или Кубиков? В случае положительного ответа выведите на экран их адреса.

Вариант 10. Известны данные о численности населения (миллионы жителей) и площади государства (тысячи квадратных километров). Создайте структуру, соответствующую условию задачи. Организуйте ввод данных согласно полям структуры. Проанализируйте данные: определите плотность населения в каждом государстве; упорядочьте государства по возрастанию плотности населения.

Вариант 11. Задайте массив структур из 16 записей, содержащий данные о сотрудниках фирмы: фамилия, имя, возраст и отношение к воинской службе (военнообязанный или нет). Организуйте ввод данных согласно полям структуры. Преобразуйте данные: распределите сотрудников по категориям «военнообязанные» и «невоеннообязанные». Выведите на экран полный список военнообязанных сотрудников.

Вариант 12. Известны данные о численности населения (миллионы жителей) и площади государства (тысячи квадратных километров). Создайте структуру, соответствующую условию задачи. Организуйте ввод данных согласно полям структуры. Задайте числовые интервалы для категорий «низкая плотность», «средняя плотность» и «высокая плотность» населения. Преобразуйте данные: распределите страны по категориям.

Вариант 13. Известны данные о росте 15 юношей группы (фамилия, имя, рост (см)). Создайте структуру, соответствующую условию задачи. Организуйте ввод данных согласно полям структуры. Упорядочьте данные по критерию убывания роста юношей. В начале нового учебного года в группу поступил новый студент. Вставьте новые данные в уже упорядоченную последовательность данных. Учтите тот факт, что рост некоторых юношей может быть одинаковым. Проанализируйте данные: определите и выведите фамилии всех юношей, рост которых меньше роста нового студента.

Вариант 14. Известны данные о росте 15 юношей группы (фамилия, имя, рост (см)). Создайте структуру, соответствующую условию задачи. Организуйте ввод данных согласно полям структуры. Упорядочьте данные по критерию убывания роста юношей. В начале нового учебного года в группу поступил новый студент. Вставьте новые данные в уже упорядоченную последовательность данных. Про-

анализируйте данные: определите и выведите фамилию юноши, рост которого меньше (больше) всего отличается от роста нового студента.

Вариант 15. Известна информация о 25 событиях, произошедших в 2000-е гг.: год, номер месяца и число. Создайте структуру, соответствующую условию задачи. Организуйте ввод данных согласно полям структуры. Проанализируйте данные. Напишите программу, сравнивающую два любых события по времени, т. е. определяющую, какое событие произошло позже.

Вариант 16. Известны данные о росте 15 юношей группы (фамилия, имя, рост (см)). Юношей одинакового роста нет. Создайте структуру, соответствующую условию задачи. Организуйте ввод данных согласно полям структуры. Упорядочьте данные в алфавитном порядке по фамилии юношей. В начале нового учебного года в группу поступил новый студент. Известно, что его рост не совпадает с ростом ни одного юноши и превышает рост самого низкого ученика. Вставьте новые данные в уже упорядоченную последовательность данных. Проанализируйте данные: определите и выведите фамилию юноши, после которого следует записать фамилию нового студента.

2.8. Массивы ячеек

Элементы массива ячеек – ячейки. Это своеобразные контейнеры, содержащие в себе любые типы массивов, включая массивы ячеек. Отличительный атрибут массивов ячеек – задание содержимого последних в фигурных скобках {}. Специальные функции для работы с массивами ячеек представлены ниже.

cell	Создает массив ячеек
cell2mat	Преобразует массив ячеек в обычный массив базового типа данных
cell2struct	Преобразует массив ячеек в массив структур
cell2table	Преобразует массив ячеек в таблицу
celldisp	Отображает содержимое массива ячеек
cellfun	Применяет функцию к каждой ячейке в массиве ячеек
cellplot	Графически отображает структуру массива ячеек
cellstr	Преобразует символьные векторы в массив ячеек
iscell	Определяет, является ли введенный массив массивом ячеек

iscellstr	Определяет, является ли введенный массив ячеек массивом ячеек из символьных векторов
mat2cell	Преобразует массив в массив ячеек, ячейки которого содержат подмассивы
num2cell	Преобразует массив в массив ячеек с последовательно отсортированными по размеру ячейками
strjoin	Объединяет в одну строку элементы строкового массива или массива ячеек из символьных векторов
strsplit	Преобразует строку или вектор символов в массив ячеек
struct2cell	Преобразует структуру в массив ячеек
table2cell	Преобразует таблицу в массив ячеек

Рассмотрим некоторые из перечисленных функций.

Функция **cell** создает массив ячеек. Массивы ячеек обычно содержат массивы символов, комбинации символов и чисел или числовые массивы различных размеров. Для работы с этим типом данных важно различать такие действия, как обращение и доступ к содержимому ячеек. Обратиться к наборам ячеек можно путем включения индексов в круглые скобки (). Доступ к содержимому ячеек выполняется путем индексации с фигурными скобками {}.

Синтаксис соедующий:

C = cell (n)
C = cell (sz1, ..., szN)
C = cell (sz)
D = cell (obj)

Команда **C = cell (n)** возвращает массив ячеек пустых матриц с размерами $n \times n$.

Команда **C = cell (sz1, ..., szN)** возвращает массив ячеек sz1, ..., szN пустых матриц, где szi – размер каждой размерности. Например, cell(2,3) возвращает массив ячеек с размерами 2 × 3.

Команда **C = cell (sz)** возвращает массив ячеек пустых матриц, где вектор размером sz задает размер массива *C*. Например, cell ([2 3]) возвращает массив ячеек с размерами 2 × 3.

Команда **D = cell (obj)** преобразует массивы, созданные в языках Java, Python, в массив ячеек MATLAB.

Создадим массив ячеек пустых матриц с размерами 3 × 3:

C = cell(3)
C=3×3 cell array

```
    {0x0 double}    {0x0 double}    {0x0 double}
    {0x0 double}    {0x0 double}    {0x0 double}
    {0x0 double}    {0x0 double}    {0x0 double}
```

Создадим массив ячеек пустых матриц с размерами 3 × 4 × 2:

```
C = cell(3,4,2);
size(C)
ans = 1×3
     3     4     2
```

Зададим значение элемента C{1,1,1} равным 6. Выведем значение на экран:

```
C = cell(3,4,2);
C{1,1,1}=6;
disp(C{1,1,1});
 >> cell_array
     6
```

Создадим одномерный массив ячеек размером 6 с помощью конструктора ячеек, используя для этого фигурные скобки. Будем хранить в нем численные скаляры – элементы C(1), C(2) и C(3), строку, трехмерный массив с размерами 5 × 10 × 2, содержащий случайные числа из интервала [0; 1], и логическое значение.

```
C = {1,2,3; 'text', rand(5, 10, 2), false}
C =
  2×3 cell array
    {[   1]}    {[         2]}    {[3]}
    {'text'}    {5×10×2 double}    {[0]}
```

Выведем содержимое ячеек на экран:

```
>> categ_array
C{1,1} =1
C{2,1} =text
C{1,2} =2
C{2,2} =
(:,:,1) =
    0.0975    0.1576    0.1419    0.6557    0.7577    0.7060    0.8235    0.4387
0.4898    0.2760
    0.2785    0.9706    0.4218    0.0357    0.7431    0.0318    0.6948    0.3816
0.4456    0.6797
    0.5469    0.9572    0.9157    0.8491    0.3922    0.2769    0.3171    0.7655
0.6463    0.6551
```

```
   0.9575    0.4854    0.7922    0.9340    0.6555    0.0462    0.9502    0.7952
0.7094    0.1626
   0.9649    0.8003    0.9595    0.6787    0.1712    0.0971    0.0344    0.1869
0.7547    0.1190
(:,:,2) =
   0.4984    0.7513    0.9593    0.8407    0.3500    0.3517    0.2858    0.0759
0.1299    0.1622
   0.9597    0.2551    0.5472    0.2543    0.1966    0.8308    0.7572    0.0540
0.5688    0.7943
   0.3404    0.5060    0.1386    0.8143    0.2511    0.5853    0.7537    0.5308
0.4694    0.3112
   0.5853    0.6991    0.1493    0.2435    0.6160    0.5497    0.3804    0.7792
0.0119    0.5285
   0.2238    0.8909    0.2575    0.9293    0.4733    0.9172    0.5678    0.9340
0.3371    0.1656
 C{1,3} =3
 C{2,3} =   0
```

Как видно из примеров, MATLAB отображает величины в каждой ячейке как значения в квадратных скобках. Обратиться к элементу массива ячеек можно как к элементу обычного массива, указав имя массива и индекс элемента в фигурных скобках.

Пример. Выведем на экран значения первого и четвертого элементов массива ячеек:

```
C{1}
ans =1
C{4}
ans(:,:,1) =
0.0336    0.4076    0.3251    0.0908    0.5271    0.6377    0.6718    0.6678
0.0067    0.4624
   0.0688    0.8200    0.1056    0.2665    0.4574    0.9577    0.6951    0.8444
0.6022    0.4243
   0.3196    0.7184    0.6110    0.1537    0.8754    0.2407    0.0680    0.3445
0.3868    0.4609
   0.5309    0.9686    0.7788    0.2810    0.5181    0.6761    0.2548    0.7805
0.9160    0.7702
   0.6544    0.5313    0.4235    0.4401    0.9436    0.2891    0.2240    0.6753
0.0012    0.3225
ans(:,:,2) =
```

0.7847 0.4735 0.7384 0.1887 0.5466 0.6358 0.1194 0.7703 0.8329 0.8699
0.4714 0.1527 0.2428 0.2875 0.4257 0.9452 0.6073 0.3502 0.2564 0.2648
0.0358 0.3411 0.9174 0.0911 0.6444 0.2089 0.4501 0.6620 0.6135 0.3181
0.1759 0.6074 0.2691 0.5762 0.6476 0.7093 0.4587 0.4162 0.5822 0.1192
0.7218 0.1917 0.7655 0.6834 0.6790 0.2362 0.6619 0.8419 0.5407 0.9398

Полностью вывести на экран значения всех элементов массива можно с помощью функции **celldisp()**. Например:

```
celldisp(C)
C{1,1} =1
C{2,1} = text
C{1,2} =2
C{2,2} =
 (:,:,1) =
0.0336 0.4076 0.3251 0.0908 0.5271 0.6377 0.6718 0.6678 0.0067 0.4624
0.0688 0.8200 0.1056 0.2665 0.4574 0.9577 0.6951 0.8444 0.6022 0.4243
0.3196 0.7184 0.6110 0.1537 0.8754 0.2407 0.0680 0.3445 0.3868 0.4609
0.5309 0.9686 0.7788 0.2810 0.5181 0.6761 0.2548 0.7805 0.9160 0.7702
0.6544 0.5313 0.4235 0.4401 0.9436 0.2891 0.2240 0.6753 0.0012 0.3225
(:,:,2) =
0.7847 0.4735 0.7384 0.1887 0.5466 0.6358 0.1194 0.7703 0.8329 0.8699
0.4714 0.1527 0.2428 0.2875 0.4257 0.9452 0.6073 0.3502 0.2564 0.2648
0.0358 0.3411 0.9174 0.0911 0.6444 0.2089 0.4501 0.6620 0.6135 0.3181
0.1759 0.6074 0.2691 0.5762 0.6476 0.7093 0.4587 0.4162 0.5822 0.1192
```

0.7218 0.1917 0.7655 0.6834 0.6790 0.2362 0.6619 0.8419 0.5407 0.9398

C{1,3} = 3

C{2,3} =0

Можно создать пустой массив ячеек и заполнить его позднее с помощью операции присваивания, как в примере ниже.

Пример. Создадим двумерный массив ячеек с размерами 2 × 2 и зададим значения некоторым его элементам:

```
c=cell(2,3)
c =
  2×3 cell array
    {0×0 double}    {0×0 double}    {0×0 double}
    {0×0 double}    {0×0 double}    {0×0 double}
```

```
>> c{1,3}=4;
>> c{1,1}=-3
c =
  2×3 cell array
    {[     -3]}    {0×0 double}    {[      4]}
    {0×0 double}    {0×0 double}    {0×0 double}
```

В практике работы со сложными данными часто возникает необходимость преобразования их типов. Эту задачу в MATLAB решает целый ряд функций, перечисленных ранее.

В качестве примера рассмотрим функцию **num2cell ()**, преобразующую обычный двумерный массив *A* с размерами 3 × 2, содержащий скаляры, в специальный массив ячеек *A*:

```
A=[1 2; 9 22; 31 6]
    A =
        1     2
        9    22
       31     6
    num2cell(A,2)
    ans =
      3×1 cell array
        {1×2 double}
        {1×2 double}
        {1×2 double}
```

Массивы ячеек используют тогда, когда имеются данные разного типа и размера и необходимо обращаться к ним как к элементам обычного массива.

Вопросы для самоконтроля

1. Чем массивы ячеек отличаются от расписания, таблиц, datetime-массивов, категориальных массивов?
2. Как создать массив ячеек?
3. Можно ли преобразовать массив ячеек в таблицу, расписание, категориальный массив? Как это сделать? Приведите пример. Когда необходимы такие преобразования? Приведите примеры.
4. Какую важную для анализа данных информацию можно получить с помощью специальных функций? Перечислите их. Приведите примеры использования.

Практическая работа

МАССИВЫ ЯЧЕЕК

Задание. Напишите программу. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий содержанию задания;

○ результат работы программы.

Варианты заданий

Вариант 1. Задайте массив ячеек, содержащий сведения о работниках предприятия: фамилия (16 символов), имя (12 символов), отчество (20 символов), должность (20 символов), год рождения (число), заработная плата (массив с размерами 12 × 12, элементы которого – зарплата за конкретный месяц). Организуйте ввод данных согласно таблице. Проанализируйте данные, выведите: среднюю заработную плату за год каждого работника; среднюю заработную плату по предприятию; список работников с наибольшим и наименьшим отклонением от нее.

№	ФИО	Должность	Год рождения	Заработная плата		
				Январь	Февраль	…

Вариант 2. Багаж пассажира характеризуется количеством (число) и списком вещей (массив с размерами 10 × 10, содержащий наименование вещи и ее вес). Имеется информация о багаже 10 пассажиров. Задайте массив ячеек, соответствующий условию задачи. Организуйте ввод данных. Проанализируйте данные: подсчитайте и сохраните в виде отдельного поля общее количество и вес вещей каждого пассажира. Выясните, имеются ли пассажиры, багаж которых превышает разрешенный лимит в 25 кг. Рассчитайте дополнительную плату для каждого пассажира из этой категории: 1 кг – 5 руб.

Вариант 3. Задайте массив ячеек из 25 записей, содержащий следующие сведения: фамилия (16 символов), имя (12 символов), адрес (массив структур, полями которого являются названия города, улицы, номер дома и квартиры) и номер телефона (12 символов). Организуйте ввод данных согласно условию задачи. Проанализируйте данные: найдите и выведите на экран фамилии и адреса людей, проживающих в городе Владимире, чей телефон начинается с цифры 3.

Вариант 4. Задайте массив ячеек, содержащий данные о 20 сотрудниках фирмы. Каждая ячейка представляет собой структуру с полями: фамилия (16 символов), имя (12 символов), отчество (20 символов), должность (20 символов), год рождения (число), пол (один символ). Организуйте ввод данных согласно условию задачи. Проанализируйте данные: определите фамилии мужчин старше 65 лет; фамилии женщин младше 55 лет.

Вариант 5. Задайте массив ячеек, содержащий данные о 16 сотрудниках фирмы. Каждая ячейка представляет собой структуру с полями: фамилия (16 символов), имя (12 символов), возраст (число), отношение к воинской службе (военнообязанный или нет). Организуйте ввод данных согласно условию задачи. Проанализируйте данные: определите разницу в возрасте между самым младшим и самым старшим по возрасту военнообязанным; фамилии людей среди невоеннообязанных, имеющих такую же разницу в возрасте.

Вариант 6. Заданы названия 26 городов и стран, в которых они находятся. Создайте массив ячеек, соответствующий условию задачи. Организуйте ввод данных. Преобразуйте данные: распределите горо-

да по категориям – названиям стран. Проанализируйте данные: вычислите процентное соотношение количества городов каждой страны.

Вариант 7. Заданы названия 26 городов и стран, в которых они находятся. Создайте массив ячеек, соответствующий условию задачи. Организуйте ввод данных. Преобразуйте данные: распределите города по категориям – «начинается с буквы А», «начинается с буквы В», «начинается с буквы С» и т. д. Проанализируйте данные: вычислите процентное соотношение количества городов, начинающихся на соответствующую букву.

Вариант 8. Известны данные о названии страны, численности населения в ней (миллионы жителей) и площади (тысячи квадратных километров). Создайте массив ячеек, соответствующий условию задачи. Организуйте ввод данных. Проанализируйте данные: определите плотность населения в каждом государстве; названия государств с наименьшей плотностью населения и их процент от общего количества.

Вариант 9. Задайте массив ячеек, содержащий данные о 30 сотрудниках фирмы: фамилия (20 символов), имя (12 символов), домашний адрес (30 символов), дата отпуска (дд: мм). Организуйте ввод данных согласно условию задачи. Распределите данные по категориям, соответствующим номеру месяца ухода сотрудника в отпуск.

Вариант 10. Известны данные о названии страны, численности населения (миллионы жителей) и площади (тысячи квадратных километров). Создайте массив ячеек, соответствующий условию задачи. Организуйте ввод данных. Проанализируйте данные: определите плотность населения в каждом государстве; упорядочьте названия государств по возрастанию плотности.

Вариант 11. Задайте массив ячеек из 16 записей, содержащий данные о сотрудниках фирмы: фамилия (15 символов), имя (12 символов), возраст (целое число) и отношение к воинской службе (военнообязанный или нет). Организуйте ввод данных согласно условию задачи. Преобразуйте данные: распределите сотрудников по категориям «военнообязанные» и «невоеннообязанные». Выведите на экран полные списки военнообязанных и невоеннообязанных сотрудников.

Вариант 12. Известны данные о названии страны, численности населения (миллионы жителей) и площади (тысячи квадратных километров). Создайте массив ячеек, соответствующий условию задачи.

Организуйте ввод данных. Задайте числовые интервалы для категорий «низкая плотность», «средняя плотность» и «высокая плотность» населения. Преобразуйте данные: распределите страны по категориям.

Вариант 13. Известны данные о 15 юношах группы: фамилия (15 символов), имя (12 символов), рост (см) при поступлении в группу и на текущий момент. Создайте массив ячеек, соответствующий условию задачи. Организуйте ввод данных. Упорядочьте данные по критерию убывания роста юношей (на текущий момент). В начале нового учебного года в группу поступил новый студент. Вставьте новые данные в уже упорядоченную последовательность данных. Учтите тот факт, что значения роста представляют собой содержимое одной ячейки. Проанализируйте данные: определите и выведите фамилии всех юношей, рост которых меньше роста нового юноши.

Вариант 14. Известны данные о 15 студентах группы: фамилия (15 символов), имя (12 символов), рост (см) при поступлении в группу и на текущий момент, пол (ж/м). Создайте массив ячеек, соответствующий условию задачи. Организуйте ввод данных. Учтите тот факт, что значения роста представляют собой содержимое одной ячейки. Распределите данные по категориям «юноши», «девушки». Проанализируйте данные: определите и выведите фамилии самых высоких юноши и девушки.

Вариант 15. Известна информация о 25 событиях, произошедших в 2000-е гг.: год, номер месяца и число, название события. Создайте массив ячеек, соответствующий условию задачи. Организуйте ввод данных. Учтите тот факт, что значения даты представляют собой содержимое одной ячейки. Проанализируйте данные, напишите программу, выводящую на экран список событий, произошедших в заданный интервал времени.

Вариант 16. Известны данные о росте 15 юношей группы: фамилия (15 символов), имя (12 символов), рост (см). Юношей одинакового роста нет. Создайте массив ячеек, соответствующий условию задачи. Организуйте ввод данных. Упорядочьте данные в алфавитном порядке по фамилии юношей. В начале нового учебного года в группу поступил новый студент. Известно, что его рост не совпадает с ростом ни одного юноши и превышает рост самого низкого ученика. Вставьте новые данные в уже упорядоченную последовательность данных. Проанализируйте данные: определите и выведите фамилию юноши, после которого следует записать фамилию нового студента.

2.9. Временные ряды

Совокупность величин {*x*1, *x*2, …, *xn*}, представляющая собой значения какого-либо параметра, изменяющегося во времени, называется временным рядом, при этом каждое значение соответствует значению параметра в конкретное время *t*1 , *t*2 , …, *tn*.

Временные ряды представляют эволюцию во времени какого-то динамического процесса, например рост населения, ситуации на рынке недвижимости и т. д.

Важно понимать различие между значением данных и выборкой данных. *Значение данных* – это одно скалярное значение, зарегистрированное в определенное время. *Выборка данных* состоит из одного или нескольких значений, сопоставленных с определенным временем.

Объекты временных рядов в MATLAB могут быть двух типов:

– timeseries – хранит информацию о выборке данных;

– tscollection – хранит набор timeseries-объектов, которые совместно используют общий временной вектор, удобный для выполнения операций на синхронизируемых временных рядах с различными модулями.

Создать timeseries-объект можно одним из способов, указанных ниже. Синтаксис следующий:

```
ts = timeseries(datavals);
ts = timeseries (datavals, timevals);
ts = timeseries(datavals, timevals, quality);
ts = timeseries(___, 'Name', tsname);
ts = timeseries ();
ts = timeseries (tsname).
```

Команда **ts = timeseries (datavals)** возвращает timeseries-объект, содержащий данные в datavals, где datavals – выборочные данные (скаляр, вектор или многомерный массив).

Команда **ts = timeseries (datavals, timevals)** возвращает timeseries-объект, содержащий данные в datavals в соответствии с временем, указанным в timevals; timevals – шаги расчета (скаляр или вектор).

Команда **ts = timeseries (datavals, timevals, quality)** задает качественные описания в терминах кодов, заданных в quality. Значениями по умолчанию являются скаляр, или вектор, или многомерный массив целых чисел в пределах от –128 до 127. Когда качественное кодовое обозначение – вектор, оно должно иметь ту же длину, что и времен-

ной вектор. Каждый элемент применяется к соответствующей выборке данных. Если качественное кодовое обозначение – массив, то оно должно иметь тот же размер, что и массив данных. Каждый элемент применяется к соответствующему элементу массива данных.

Команда **ts = timeseries (___, 'Name', tsname)** задает имя tsname для timeseries-объекта.

Команда **ts = timeseries ()** возвращает пустой timeseries-объект.

Команда **ts = timeseries (tsname)** создает пустой timeseries-объект с именем tsname.

Допустимые векторы символов даты могут иметь формы, представленные ниже.

Формат	*Пример*
dd-mmm-yyyy HH:MM:SS	01-Mar-2000 15:45:17
dd-mmm-yyyy	01-Mar-2000
mm/dd/yy	03/01/00
mm/dd	03/01
HH:MM:SS	15:45:17
HH:MM:SS PM	3:45:17 PM
HH:MM	15:45
HH:MM PM	3:45 PM
mmm.dd,yyyy HH:MM:SS	Mar.01,2000 15:45:17
mmm.dd,yyyy	Mar.01,2000
mm/dd/yyyy	03/01/2000

Пример. Создадим timeseries-объект, содержащий пять выборочных данных, заданных с интервалом в 10 единиц измерения времени:

```
ts3 = timeseries((1:5)',[0 10 20 30 40]);
ts3
Common Properties:
            Name: 'unnamed'
            Time: [5x1 double]
        TimeInfo: [1x1 tsdata.timemetadata]
            Data: [5x1 double]
        DataInfo: [1x1 tsdata.datametadata]
  More properties, Methods
```

Пример. Создадим timeseries-объект с пятью выборками данных, где каждая выборка – вектор-столбец длиной 2:

```
ts2 = timeseries(rand(2,5))
```

```
Common Properties:
            Name: 'unnamed'
            Time: [2x1 double]
        TimeInfo: [1x1 tsdata.timemetadata]
            Data: [2x5 double]
        DataInfo: [1x1 tsdata.datametadata]
```

Пример. Создадим timeseries-объект с шестью выборочными данными в виде массива, заданными с интервалом в одну единицу измерения времени:

```
y=[15000 10000 20000 30000 40000 24000];
x=[1 2 3 4 5 6];
ts = timeseries(x, y);
disp(ts)
Events: []
               Name: 'unnamed'
           UserData: []
               Data: [1×1×6 double]
           DataInfo: [1×1 tsdata.datametadata]
               Time: [6×1 double]
           TimeInfo: [1×1 tsdata.timemetadata]
            Quality: []
        QualityInfo: [1×1 tsdata.qualmetadata]
         IsTimeFirst: 0
    TreatNaNasMissing: 1
             Length: 6
```

Вопросы для самоконтроля

1. Чем временные ряды отличаются от расписания, таблиц, структур, datetime-массивов, массивов ячеек и категориальных массивов?

2. Как создать временной ряд?

3. Можно ли преобразовать временной ряд в массив ячеек, таблицу, расписание, структуру, категориальный массив? Как это сделать? Приведите пример. Когда необходимы такие преобразования? Приведите примеры.

4. Какую важную для анализа данных информацию можно получить с помощью специальных функций? Перечислите их. Приведите примеры использования.

Практическая работа

ВРЕМЕННЫЕ РЯДЫ

Задание. Напишите программу. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий содержанию задания;

○ результат работы программы.

Варианты заданий

Вариант 1. Задан ряд показателей урожайности зерновых культур в целом по России (ц/га) за 1984 – 1998 гг.: 15,6; 17,6; 16,4; 15,6; 17,6; 20,3; 15,8; 18,8; 17,9; 15,6; 12,5; 14,0; 17,8; 10,4; 10,6. Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и выявите года: а) с наименьшими (наибольшими) показателями урожайности; б) в течение которых урожайность снижалась (росла).

Вариант 2. Дана таблица значений некоторой функциональной зависимости, полученной из 10 опытов, проведенных в течение первой декады декабря 2021 г.

x	1	2	3	4	5	6	7	8	9	10
y	1,0	1,5	3,0	4,5	7,0	8,5	8,0	11,3	1,0	9,5

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и выявите: а) наибольшее (наименьшее) значение показателей; б) показатели, наиболее отклоняющиеся от максимального значения.

Вариант 3. В таблице приведены данные о среднедушевом прожиточном минимуме в день одного трудоспособного x (тыс. руб.) и среднедневной заработной плате y (тыс. руб.) по 12 регионам России за 2020 г.

Дата	1.01	1.02	1.03	1.04	1.05	1.06	1.07	1.08	1.09	1.10	1.11	1.12
Номер региона	1	2	3	4	5	6	7	8	9	10	11	12
x	78	82	87	79	89	106	67	88	73	87	76	115
y	133	148	134	154	162	195	139	158	152	162	159	173

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и вычислите среднее значение *y*. Выявите регионы с наибольшим (наименьшим) отклонением прожиточного минимума от среднего значения *y*.

Вариант 4. В таблице представлен список наиболее интересных моделей оперативной памяти, указаны размер памяти, цена по состоянию на 7 января 2021 г.

Модель оперативной памяти	Размер, Гб	Цена, руб.
Kingston ValueRAM KVR16N11/8	16	2820
Samsung M378A1K43CB2-CTD	8	2880
ПК HyperX Fury HX432C16FB3K2/8	16	3480
Corsair CMSA4GX3M1A1066C7	4	1789
Crucial CT8G4DFS824A	8	2885
G.SKILL Ripjaws V F4-3200C16D-16GVKB	16	6600
HyperX Fury HX316C10FB/8	8	3772
Kingston ValueRAM KVR26S19S8/8	8	2600
AMD R538G1601S2S-UO	8	2490
G.SKILL Trident Z RGB F4-3200C14D-32GTZR	32	22 740
Samsung M378A4G43MB1-CTD	32	10 858
Thermaltake TOUGHRAM RGB R009D408GX2-4400C19A	16	14 210
Crucial CT8G4DFS824A	8	2600

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Упорядочьте данные по размеру оперативной памяти. Проанализируйте данные и вычислите среднюю цену для каждого размера оперативной памяти.

Вариант 5. В таблице представлены данные зависимости тока от мощности и частоты вращения двигателя на момент проведения эксперимента (любая дата).

Мощность, кВт	Среднее значение токов холостого хода (в долях от силы номинального тока) при синхронной частоте вращения, об/мин				
	3000	1500	1000	750	600
0,5 – 1	0,4	0,55	0,6	–	–
1,1 – 5	0,35	0,5	0,55	0,6	–
5,1 – 10	0,25	0,45	0,5	0,55	0,6
10,1 – 25	0,2	0,4	0,45	0,5	0,55
25,1 – 50	0,18	0,35	0,4	0,45	0,5

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и выявите, для каких значений частоты вращения и мощности двигателя среднее значение токов холостого хода наименее отличается от максимального значения.

Вариант 6. Данные рейтинга субъектов Федерации по средней выручке за 2019 г. с сайта https://www.spark-interfax.ru/statistics/ представлены ниже.

Субъект Федерации	Средняя выручка, млн руб.
Москва	117,17
Санкт-Петербург	100,19
Московская область	82,03
Тюменская область	156,35
Свердловская область	63,43
Краснодарский край	57,85
Республика Татарстан	54,55
Нижегородская область	71,60
Самарская область	55,52
Республика Башкортостан	59,41

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и вычислите общую среднюю выручку, значение разности выручки каждого субъекта Федерации от общего среднего значения.

Вариант 7. Данные рейтинга субъектов Федерации по количеству юридических лиц (ЮЛ) за 2019 г. с сайта https://www.spark-interfax.ru/statistics представлены ниже.

Субъект Федерации	Количество компаний
Москва	711 017
Санкт-Петербург	279 544
Московская область	206 395
Тюменская область	80 213
Свердловская область	126 322
Краснодарский край	116 358
Республика Татарстан	107 415
Нижегородская область	81 270
Самарская область	97 637
Новосибирская область	1025

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Отсортируйте данные по значениям столбца «Количество компаний». Проанализируйте данные и выведите на экран субъекты Федерации с количеством компаний ниже порогового уровня в 90 000.

Вариант 8. Данные рейтинга субъектов Федерации по количеству компаний в области энергетики за 2019 г. с сайта https://www.spark-interfax.ru/statistics представлены ниже.

Субъект Федерации	Количество компаний
Москва	1500
Санкт-Петербург	741
Московская область	1190
Тюменская область	509
Свердловская область	700
Краснодарский край	534
Челябинская область	548
Нижегородская область	543
Красноярский край	534
Новосибирская область	587

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Отсортируйте данные по значениям столбца «Количество компаний». Проанализируйте данные и выведите на экран субъекты Федерации с количеством компаний выше порогового уровня, равного минимальному значению количества компаний.

Вариант 9. Данные о численности населения и количестве трудоспособных за период с 2006 по 2019 г. с сайта https://rosinfostat.ru/ekonomicheski-aktivnoe-naselenie/ представлены ниже.

Год	Численность населения, тыс. чел.	Занятые, тыс. чел.
2006	74 419	69 169
2007	75 289	70 770
2008	75 700	71 003
2009	75 694	69 410
2010	75 478	69 934
2011	75 779	70 857
2012	75 676	71 545
2013	75 529	71 391
2014	75 428	71 539
2015	76 588	72 324
2016	76 636	72 393
2017	76 285	72 316
2018	76 190	72 532
2019	75 398	71 933

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные: в какие годы численность населения превышала 75 000 тыс. чел.? Какой процент составляли занятые?

Вариант 10. Статистика естественного движения населения по данным Росстата (https://rosinfostat.ru/smertnost/#i-2) на сентябрь 2021 г. представлена ниже.

Наименование	Родившихся, чел.	Умерших, чел.
Российская Федерация	1 604 344	1 828 910
Центральный федеральный округ	391 129	508 436
Северо-Западный федеральный округ	145 537	176 127
Южный федеральный округ	173 257	210 304
Северо-Кавказский федеральный округ	141 841	73 338
Приволжский федеральный округ	311 450	390 946
Уральский федеральный округ	147 057	146 947
Сибирский федеральный округ	196 185	224 041
Дальневосточный федеральный округ	97 888	98 721

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и вычислите естественный прирост населения в каждом федеральном округе. Определите федеральные округа с отрицательным значением прироста.

Вариант 11. На сайте Росстата представлены данные о естественном распределении населения (сельское, городское) за период с 1990 по 2019 г. (https://rosinfostat.ru/smertnost/#i-2) (рис. 2.4).

Годы	Все население			Городское			Сельское население		
	всего	муж	жен	всего	муж	жен	всего	муж	жен
1990	69,19	63,73	74,30	69,55	64,31	74,34	67,97	62,03	73,95
1995	64,52	58,12	71,59	64,70	58,30	71,64	63,99	57,64	71,40
2000	65,34	59,03	72,26	65,69	59,35	72,46	64,34	58,14	71,66
2001	65,23	58,92	72,17	65,57	59,23	72,37	64,25	58,07	71,57
2002	64,95	58,68	71,90	65,40	59,09	72,18	63,68	57,54	71,09
2003	64,84	58,53	71,85	65,36	59,01	72,20	63,34	57,20	70,81
2004	65,31	58,91	72,36	65,87	59,42	72,73	63,77	57,56	71,27
2005	65,37	58,92	72,47	66,10	59,58	72,99	63,45	57,22	71,06
2006	66,69	60,43	73,34	67,43	61,12	73,88	64,74	58,69	71,86
2007	67,61	61,46	74,02	68,37	62,20	74,54	65,59	59,57	72,56
2008	67,99	61,92	74,28	68,77	62,67	74,83	65,93	60,00	72,77
2009	68,78	62,87	74,79	69,57	63,65	75,34	66,67	60,86	73,27
2010	68,94	63,09	74,88	69,69	63,82	75,39	66,92	61,19	73,42
2011	69,83	64,04	75,61	70,51	64,67	76,10	67,99	62,40	74,21
2012	70,24	64,56	75,86	70,83	65,10	76,27	68,61	63,12	74,66
2013	70,76	65,13	76,30	71,33	65,64	76,70	69,18	63,75	75,13
2014	70,93	65,29	76,47	71,44	65,75	76,83	69,49	64,07	75,43
2015	71,39	65,92	76,71	71,91	66,38	77,09	69,90	64,67	75,59
2016	71,87	66,50	77,06	72,35	66,91	77,38	70,50	65,36	76,07
2017	72,70	67,51	77,64	73,16	67,90	77,96	71,38	66,43	76,66
2018	72,91	67,75	77,82	73,34	68,11	78,09	71,67	66,75	76,93
2019	73,34	68,24	78,17	73,72	68,56	78,41	72,21	67,36	77,39

Рис. 2.4. Данные о распределении населения

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и вычисли-

те, какой процент от общей численности населения составляют: а) мужчины (женщины); б) городское (сельское) население. В какие периоды численность сельского населения превышала численность городского населения?

Вариант 12. На сайте Росстата представлены данные о естественном распределении населения (сельское, городское) за период с 1990 по 2019 г. (https://rosinfostat.ru/smertnost/#i-2) (см. рис. 2.4). Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и вычислите, какой процент от городского населения составляют мужчины (женщины). В какие периоды численность мужского городского населения превышала численность женского?

Вариант 13. На сайте Росстата представлены данные о естественном распределении населения (сельское, городское) за период с 1990 по 2019 г. (https://rosinfostat.ru/smertnost/#i-2) (см. рис. 2.4). Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и вычислите, какой процент от сельского населения составляют мужчины (женщины). В какие периоды численность мужского сельского населения не превышала численность женского?

Вариант 14. На сайте Росстата представлены данные о демографической статистике за период с 1950 по 2007 г. (https://rosinfostat.ru/smertnost/#i-2).

Год	Родившихся, чел.	Умерших, чел.
1950	2 745 997	1 031 010
1970	1 903 713	1 131 183
1980	2 202 779	1 525 755
1990	1 988 858	1 655 993
1995	1 363 806	2 203 811
2000	1 266 880	2 225 332
2001	1 311 604	2 254 856
2002	1 396 967	2 332 272
2003	1 477 301	2 365 826
2004	1 502 477	2 295 402
2005	1 457 376	2 303 935
2006	1 479 637	2 166 703
2007	1 610 122	2 166 703

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и вычислите естественный прирост населения по годам. Определите тенденцию прироста за весь период – убывающая или возрастающая.

Вариант 15. На сайте Росстата представлена статистика естественных причин смертности за период с 2000 по 2019 г. (https://rosinfostat.ru/smertnost/#i-2) (рис. 2.5).

Годы	Болезни органов дыхания		Болезни органов пищеварения		Внешние причины	
	мужчины	женщины	мужчины	женщины	мужчины	женщины
2000	72230	29911	37907	26769	250009	68707
2001	68561	26361	40671	28740	259109	72525
2002	72564	28439	44046	31468	264055	75241
2003	73017	28107	47490	34028	260323	74850
2004	67777	25177	49378	35924	254109	73014
2005	69131	25605	54342	39466	246257	69658
2006	59190	23571	51400	38039	219216	63569
2007	55594	22353	50051	37629	201334	58078
2008	56863	22630	51709	38692	189552	54911
2009	55463	23999	50676	38280	173089	51487
2010	52944	21864	52500	39495	167060	49807
2011	52144	22075	50387	38523	153544	45814
2012	49199	21594	49823	39044	149104	44670
2013	50368	23700	49063	39368	143444	41909
2014	52783	25529	53743	42946	144430	42349
2015	50990	24823	56216	45740	136196	41394
2016	47574	22758	53992	44223	128411	39132
2017	42050	19982	50538	42451	116556	36185
2018	41243	19907	51551	43879	110407	34205
2019	40601	18587	52592	45679	105016	32617

Рис. 2.5. Статистика естественных причин смертности

Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и определите процент мужчин и женщин, умирающих по каждой из указанных причин. Выявите причину с наибольшим показателем смертности.

Вариант 16. На сайте Росстата представлена статистика естественных причин смертности за период с 2000 по 2019 г. (https://rosinfostat.ru/smertnost/#i-2) (см. рис. 2.5). Организуйте ввод данных согласно содержанию задания с использованием временных рядов. Проанализируйте данные и вычислите средние показатели смертности мужчин и женщин по каждой причине. Выявите причину с наименьшим средним показателем смертности.

2.10. Строковый тип

Строковое представление данных лежит в основе символьной математики, арифметики произвольной точности, работы с базами данных, файлами и массивами ячеек. Очевидно, умение работать со строками – необходимое условие корректного анализа данных.

На сегодняшний день в MATLAB реализовано два способа представления текста: в виде массива (векторов) символов (char) и в виде строковых массивов (string).

До 2016 г. в MATLAB был только один строковый тип данных – char, и строка представляла собой вектор данных этого типа. Типичное применение такого представления данных – сохранение коротких текстов в виде векторов символов, как в примере ниже:

```
s='Hello, World';
whos s;
  Name      Size            Bytes  Class    Attributes
   s        1x12               24  char
```

Как видно, MATLAB отображает векторы символов с одинарными кавычками.

Массивы строк хранились как массивы ячеек (cellarray). Следовательно, необходимо было помнить о согласовании размерности векторов типа char, что значительно осложняло работу с ними, особенно в части их разбиения, слияния и сравнения.

Тип данных string появился в MATLAB, начиная с версии R2016b, и специально предназначен для хранения строк и операций над строками.

Для того чтобы представить строку в MATLAB, произвольный текст обрамляют двойными кавычками.

Пример. Создание и инициализация строки:

```
strchar="Hello, World";
whos strchar;
```

```
>> stroka1_1
 Name      Size            Bytes  Class     Attributes
 strchar      1x1               166  string
```

Как видно, в случае вектора символов строка представляет собой массив из символов. Их ровно столько, сколько символов в строке. В этом случае для преобразования строки используются типовые алгоритмы работы с массивами, рассмотренные в главах 2 и 4. Кроме того, используется ряд специфичных функций, рассмотренных ранее.

2.10.1. Основные операции с символьными векторами

К операциям над строками обычно относят поиск вхождений одних строк в другие, замену регистров символов, объединение и преобразование строк и т. д. Следующие функции, представленные ниже, осуществляют операции над строками обоих типов.

string	Создает строковый массив
strings	Создает массив строк без символов
join	Объединяет строки
plus	Добавляет числа, строки
char	Массив символов
cellstr	Преобразует символьные векторы в массив ячеек
blanks	Создает символьный массив пробелов
newline	Создает символ новой строки
compose	Данные о формате в несколько строк
sprintf	Данные о формате в строку или вектор символов
strcat	Конкатенация строк горизонтально
append	Объединяет строки
convertCharsToStrings	Преобразует символьные массивы в строковые массивы, оставляя другие массивы неизменными
convertStringsToChars	Преобразует строковые массивы в символьные массивы, оставляя другие массивы неизменными
convertContainedStrings ToChars	Преобразует строковые массивы на любом уровне массива ячеек или структуры

double	Массивы с двойной точностью
string	Массив строк
str2double	Преобразует строки в значения двойной точности
num2str	Преобразует числа в символьный массив
ischar	Определяет, являются ли входные данные массивом символов
iscellstr	Определяет, являются ли входные данные массивом ячеек из символьных векторов
isstring	Определяет, являются ли входные данные массивом строк
isStringScalar	Определяет, являются ли входные данные массивом строк с одним элементом
strlength	Длины строк
isstrprop	Определяет, какие характеры во входных строках имеют заданную категорию
isletter	Определяет, какие символы являются буквами
isspace	Определяет, какие символы являются пробелами
contains	Определяет, находится ли шаблон в строках
matches	Определяет, совпадает ли шаблон со строками
count	Считает случаи шаблона в строках
endsWith	Определяет, заканчиваются ли строки шаблоном
startsWith	Определяет, начинаются ли строки с шаблона
strfind	Строки поиска в других строках
sscanf	Считает отформатированные данные из строк
replace	Находит и заменяет одну или несколько подстрок
replaceBetween	Заменяет подстроки между начальными и конечными точками
strrep	Находит и заменяет подстроки
pattern	Находит в тексте совпадения с шаблоном
alphanumericsPattern	Создает шаблон, соответствующий тексту, состоящему из одного или нескольких буквенных и цифровых символов

characterListPattern	Сопоставляет символы строки с символами из списка
digitsPattern	Находит в тексте совпадения с цифровым шаблоном
lettersPattern	Находит в тексте совпадения с символьным шаблоном

Рассмотрим некоторые из перечисленных функций.

Функция **string ()** представляет текст в виде массива строк (строкового скаляра). Каждый элемент массива строк хранит последовательность символов. Последовательности могут иметь различные длины, например «yes» и «no». Массив строк, состоящий из одного элемента, также является строковым скаляром.

Строковые массивы можно индексировать, изменять и конкатенировать с помощью стандартных операций над массивами. Если массив строк представляет собой числа, можно преобразовать его в числовой массив с помощью функции **double**.

Синтаксис следующий.

Команда **str = string(A)** преобразует переменные различных типов данных в строковые массивы. Например, преобразуем числовой вектор *A* в строковый массив. Для обратного преобразования используем функцию **double**:

```
A=[1 20 300];
%преобразование числового массива в строку
str=string (A);
display (str)
%преобразование строки в числовой массив
B=double (str);
whos B
display (B)
```

```
  str =
   1×3 string array
     "1"    "20"    "300"
  B         1x3              24  double
  B =
       1    20   300
```

Команда **str = string (A, dateFmt)**, где A – datetime- или duration-массив, **dateFmt** – формат, заданный как "HH:mm:ss", преобразует массив *A* в строковый массив.

Создадим duration-массив *D* и преобразуем его в массив строк:

```
D = hours(23:25) + minutes(8) + seconds(1.2345);
display (D)
%преобразование массива в строку
str=char (D);
display (str);
D =
  1×3 duration array
   23.134 hr   24.134 hr   25.134 hr
str =
  3×9 char array
    '23.134 hr'
    '24.134 hr'
    '25.134 hr'
```

Часто в задачах анализа данных требуется выполнить разделение строки и найти уникальные слова. Рассмотрим на примере, как это сделать.

Во-первых, создадим строковый скаляр:

```
str = "A horse! A horse! My kingdom for a horse!"
```

Удалим восклицательные знаки в строке:

```
str = erase(str,"!")
```

Преобразуем все буквы в строковом скаляре к единому регистру. Это необходимо, иначе две строки «AAAA» и «aaaa» система посчитает разными:

```
str = lower(str)
```

Разделим строковый скаляр на пробелы с помощью функции **split**, которая отбросит пробелы и вернет результат в виде массива строк:

```
str = split(str)
str = 9x1 string
    "a"
    "horse"
    "a"
    "horse"
```

```
   "my"
   "kingdom"
   "for"
   "a"
   "horse"
```

Найдем уникальные слова в строковом скаляре с использованием функции **unique**.

```
str = unique(str)
str = 5x1 string
   "a"
   "for"
   "horse"
   "kingdom"
   "my"
```

Полный текст программы представлен ниже:

```
str = "A horse! A horse! My kingdom for a horse!";
str = erase(str,"!");
str = lower(str);
str = split(str);
display (str);
str = unique(str);
```

Функция **strlength ()** используется для нахождения длины строки.

Команда **L = strlength(str)** возвращает количество символов в строке str. Например, подсчитаем количество символов в строке «Hello, World». Начиная с версии R2017a, можно создать строку с помощью двойных кавычек. Результатом будет массив строк с размерами 1 × 1, или строковый скаляр.

```
str = "Hello, World"
L = strlength(str)
str =
   "Hello, World"
L =
   12
```

Можно подсчитать длину каждой строки в массиве, например так:

```
str = ["Amis","Chekhov","Joyce";"Stein","","Proust"]
L = strlength(str)
2×3 string array
```

```
   "Amis"     "Chekhov"    "Joyce"
   "Stein"    ""           "Proust"
L =
    4     7     5
    5     0     6
```

Можно объединить строки и подсчитать их длину, например:

```
s1="Home";
s2="some";
s3="dogs";
s=strcat(s1,s2,s3);
disp (s);
disp (strlength (s));
Homesomedogs
   12
```

Функция **strfind** выполняет поиск индексов вхождения подстроки в строку.

Синтаксис следующий:

```
k = strfind (str, pat)
k = strfind (str, pat, 'ForceCellOutput', cellOutput)
```

Команда **k = strfind (str, pat)** обеспечивает поиск начальных индексов более короткой строки pat внутри более длинной строки str и возвращает вектор этих индексов. Индексы указывают положение первого символа pat в строке str. Если pat не найден, команда **strfind** возвращает пустой массив []. Поиск выполняется с учетом регистра.

В примере ниже более короткая строка str2 находится в строке str1, начиная с позиций 12 и 28:

```
str1='Example of the function Is the findstr function';
str2='the';
k = strfind(str1,str2);
disp(k);
>> stroka1
   12    28
```

Команда **k = strfind (str, pat, 'ForceCellOutput', cellOutput)** находит случаи вхождения подстроки pat в строковый скаляр str, возвращает начальные индексы этих случаев (cellOutput=true) в массив ячеек. В примере ниже в массив ячеек *k* будут возвращены индексы вхождения подстроки 'ain':

```
str = 'The rain in Spain.';
k = strfind(str,'ain','ForceCellOutput',true);
disp (k{1})
>> categ_array
    6    15
```

Функция **lower ('str')** возвращает строку символов str, в которой символы верхнего регистра переводятся в нижний регистр, а все остальные символы остаются без изменений. Синтаксис следующий:

```
newDocuments = lower (str)
```

Команда **newDocuments = lower (str)** преобразует в строковом скаляре str каждый символ верхнего регистра в соответствующий ему символ нижнего регистра, оставляя все другие символы без изменений:

```
str='Example Of The Function';
t=lower(str);
disp(t);
>> stroka2
    example of the function
```

Функция **upper ('str')** возвращает строковый скаляр str, в котором все символы нижнего регистра переводятся в верхний регистр, а все остальные символы остаются без изменений. Синтаксис аналогичен функции **lower**. Например:

```
str='danger!';
t = upper(str);
disp(t);
>> stroka3
DANGER!
```

Функция **strcat** выполняет горизонтальную конкатенацию строк. Синтаксис следующий:

```
s = strcat (s1, ..., sN)
```

Команда **s = strcat (s1, ..., sN)** выполняет горизонтальное объединение соответствующих рядов массивов символов s1, s2, s3 и так далее, причем пробелы в конце каждого ряда отбрасываются, и возвращает объединенную строку (ряд) результирующего массива символов, пробелы добавляются заново после анализа строк в полученном массиве. Все входные массивы должны иметь одинаковое число

строк. В частном случае они могут быть представлены в виде одной строки символов. Если один из входных аргументов – не массив символов, а строковый массив ячеек, то любой из других входных аргументов может быть скаляром или любым массивом той же размерности. Если входной массив состоит только из символов, то выходной массив также будет массивом символов. Если любой из входных массивов – строковый массив ячеек, то функция **strcat ()** возвращает строковый массив ячеек, сформированный из объединенных соответствующих элементов массивов s1, s2, s3, при этом любой из элементов может быть скаляром. Например:

```
s1='Home';
s2='some';
s3='dogs';
s=strcat (s1, s2, s3);
disp (s);
>> char5
Homesomedogs
s2{ 1.1}= 'home':
s2{ 1.2}= 'reading';
s2
s2 =
'home' 'reading'
t = strcat(s1.s2)
t =
'Homehome' 'book reading'
s1=['wri ']
s2=['ter']
t = strcat(s1.s2)
t =
writer
```

Функция **strvcat** выполняет вертикальное объединение строк. Синтаксис следующий:

```
S = strvcat (t1, t2, t3, ...)
S = strvcat (c)
```

Команда **S = strvcat (t1, t2, t3, ...)** формирует символьный массив *S*, содержащий символьные массивы t1, t2, t3, ... как строки. Про-

белы добавлены к каждому входному параметру по мере необходимости, так чтобы количество символов в строке *S* не менялось.

Например:

```
 t1=['string'];
 t2=['concatenation']:
S = strvcat(t1.t2)
S =
string concatenation
```

Функция игнорирует пустые символьные векторы, поэтому в современных версиях MATLAB рекомендуется использовать функцию **char**.

Функция **char** создает массив символов. Синтаксис следующий:

```
C = char (A)
C = char (A1, ..., An)
C = char (A, dateFmt)
```

Команда **C = char (A)** преобразует входной массив *A* к символьному массиву. Например, преобразуем числовой массив в символьный:

```
A = [77 65 84 76 65 66];
C = char(A);
disp(C)
>> categ_array
C =
   'MATLAB'
```

Как видно из примера, на экран выведены символы, ASCII-коды которых – элементы массива *A*, т. е. по коду 77 выводится символ «M», по коду 65 – символ «A» и т. д.

Команда **C = char (A1, ..., An)** преобразует массивы A1, ..., An в односимвольный массив. После преобразования в символы входные массивы становятся строками в массиве *C*. В отличие от функции **strvcat**, функция **char** заполняет строки пробелами по мере необходимости. Если какой-либо входной массив является пустым символьным массивом, то соответствующая строка в массиве *C* будет строкой пробелов. Входные массивы A1, ..., An могут иметь различные размеры, но не могут быть строковыми массивами, массивами ячеек или категориальными массивами. Например, преобразуем несколько массивов в односимвольный массив:

```
A1 = [65 66; 67 68];
A2 = 'abcd';
C = char(A1,A2);
disp(C)
>> categ_array
AB
CD
abcd
```

Как видно из примера, результирующий массив *C* имеет размер 3 × 2, первые две строки которого – элементы массива A1, третья строка – элементы массива A2.

Команда **C = char (A, dateFmt)**, где A – datetime- или duration-массив, преобразует массив *A* в символьный с применением заданного формата dateFmt вида "HH:mm:ss".

Например, создадим duration-массив. Преобразуем массив *D* в символьный массив, который представляет одно значение длительности по времени:

```
D = hours(23:25) + minutes(8) + seconds(1.2345)
C = char(D)
D = 1x3 duration
   23.134 hr   24.134 hr   25.134 hr
C = 3x9 char array
    '23.134 hr'
    '24.134 hr'
    '25.134 hr'
```

Функция **strcmp** сравнивает строки. Синтаксис следующий:

```
tf = strcmp ('str1', 'str2')
```

Команда **tf = strcmp ('str1', 'str2')** возвращает логическую единицу, если две сравниваемые строки str1 и str2 идентичны, и логический ноль – в противном случае. Например:

```
str1='computer';
str2='computer';
k = strcmp(str1,str2);
display (k);
  logical
   1
S1='first';
```

```
S2='second';
T='third';
TF = strcmp(S1,T);
display (TF);
logical
  0
```

Возможен вид **tf = strcmp (S,T)**. В этом случае возвращается строковый массив ячеек tf, содержащий единицы для идентичных элементов массивов *S* и *T* и нули для всех остальных. Сравним каждый элемент в двух массивах ячеек из символьных векторов:

```
s = {'Time','flies','when';
     'you''re','having','fun.'};
t= {'Time','drags','when';
     'you''re','anxiously','waiting.'};
tf = strcmp(s,t)
tf = 2x3 logical array
  1   0   1
  1   0   0
```

Начиная с версии R2017a, можно создавать строки с помощью двойных кавычек. Например:

```
s1 = ["A","bc";
     "def","G"];
s2 = ["B","c";
     "def","G"];
tf = strcmp(s1,s2)
tf = 2x2 logical array
     0   0
     1   1
```

Если один из массивов – строковый массив ячеек, то предварительно из массива символов удаляются пробелы в конце строк. Массивы *S* и *T* должны иметь одинаковый размер.

Функция **strncmp ('str1', 'str2', n)** возвращает логическую единицу, если две сравниваемые строки str1 и str2 содержат *n* первых идентичных символов, и логический ноль – в противном случае. Аргументы str1 и str2 могут быть также строковыми массивами ячеек.

Команда **tf = strncmp (S, T, n)** возвращает строковый массив ячеек tf, содержащий единицы для идентичных (до *n* символов) элементов массивов *S* и *T* и нули для всех остальных. Например:

```
str1='computer';
str2='computer for me';
k = strncmp(str1,str2, 3);
display (k);
k = strncmp(str1,str2,12);
display (k);
k =
  logical
   1
k =
  logical
   0
```

Функция **strrep** осуществляет поиск и замену подстрок. Синтаксис следующий:

```
newStr = strrep (str, old, new)
```

Функция **strrep (str, old, new)** заменяет все подстроки old, найденные внутри строки символов str, на строку new. Если какой-либо входной параметр – не скалярный массив строк или массив ячеек из символьных векторов, то другие входные параметры должны иметь совместимые размеры. Например:

```
chr = 'The quick brown fox'
newChr = strrep(chr,'quick','sly')
disp (newChar)
    newChr =
'The sly brown fox'
```

Можно заменить текст в массиве строк. В примере ниже создан массив строк str. Начиная с версии R2017a, можно создавать строки с помощью двойных кавычек. В каждом элементе массива заменим подстроку ‘the’ на подстроку ‘a’:

```
str = ["the quick brown fox";
      "and the lazy dog"]
newStr = strrep(str,'the','a')
newStr = 2x1 string
```

"a quick brown fox"
"and a lazy dog"

Подстроки можно заменять на несколько значений. В примере ниже значения '___' в первом массиве ячеек заменены различными значениями второго массива ячеек:

```
1 = {'Date Received: ___';
     'Date Accepted: ___'};
old = '___';
new = {'2016-09-06';
       '2016-10-11'};
C2 = strrep(C1,old,new)
C2 = 2x1 cell
   {'Date Received: 2016-09-06'}
   {'Date Accepted: 2016-10-11'}
```

Функция **num2str** преобразует числа в символьный массив. Синтаксис следующий:

```
s = num2str (A)
s = num2str (A, precision)
s = num2str (A, formatSpec)
```

Команда **s = num2str (A)** преобразует элементы числового массива *A* в символьный массив. Аргумент *A* может быть скаляром, вектором или матрицей. В анализе данных такое преобразование полезно для графиков маркировки и создания заголовков с числовыми значениями.

В примере ниже π отображено как число с плавающей запятой заданной точности:

```
s = num2str(pi)
s =
   '3.1416'
```

Команда **s = num2str (A, precision)** задает количество значащих цифр для элементов массива *A*. Например, зададим максимальное количество значащих цифр для значений с плавающей запятой, равных 3:

```
%генерация случайных чисел
rng('default')
%массив случайных чисел размером 2*2
A = randn([2,2]);
disp(A)
```

```
%задание точности для элементов массива
s = num2str(A,3)
disp(s)
>> categ_array
   0.5377   -2.2588
   1.8339    0.8622
0.538     -2.26
 1.83     0.862
```

2.10.2. Функции преобразования систем счисления

Некоторые строковые функции служат для преобразования систем счисления.

Функция **bin2dec ('binarystr')** возвращает десятичное число, эквивалентное строке двоичных символов binarystr. Например:

```
bin2dec ('101')
ans =
5
```

Функция **dec2bin (d)** возвращает строку двоичных символов (0 и 1), эквивалентную десятичному числу *d*. Аргумент *d* должен быть неотрицательным целым числом меньше 2^{52}. Например, выведем двоичное представление десятичного числа 512:

```
s=dec2bin (512)
s =
   '1000000000'
```

Функция **dec2bin (d, n)** возвращает строку двоичных символов, содержащую по меньшей мере *n* бит. Например, выведем двоичное представление десятичного числа 512 с точностью 14 знаков:

```
s=dec2bin (512, 14)
s =
   '00001000000000'
```

Функция **dec2base (d, n)** возвращает строку символов, представляющих десятичное число *d* как число в системе счисления с основанием *n*. Например:

```
str = dec2base (1234,16)
str =
          4D2
```

Функция **dec2hex (d)** возвращает шестнадцатеричную строку символов, эквивалентную числу *d*. Аргумент *d* должен быть неотрицательным целым числом меньше 2^{52}.

Команда **str = dec2hex (d.n)** возвращает шестнадцатеричную строку, содержащую по меньшей мере *n* цифр. Например:

```
str = dec2hex(1234)
str =
        4D2
```

Функция **base2dec (S, B)** преобразует строку символов *S*, представляющих число в системе счисления по основанию *B*, в символьное представление десятичного числа. Например:

```
d = base2dec('4D2',16)
d =
        1234
```

Функция **hex2dec ('hex_value')** возвращает число *d*, представленное строкой шестнадцатеричных символов hex_value. Если аргумент hex_value – массив символов, то каждая строка этого массива интерпретируется как шестнадцатеричное представление числа. Например:

```
d = hex2dec('4D2')
d =
   1234
```

Функция **hex2num ('hex_value')** возвращает десятичное число *f* с удвоенной точностью, эквивалентное шестнадцатеричному числу, находящемуся в строке символов hex_value. Например:

```
f = hex2num ('4831fb52a18')
f =
    6.1189e+039
```

2.10.3. Вычисление значения строки

Строковые выражения обычно не вычисляются, так что, к примеру, вывод строки '2+3' просто повторяет строку:

```
str='2+3';
display (str);
>> stroka
str =
   '2+3'
```

Однако с помощью функции **eval** ('строковое выражение') строка, представляющая математическое выражение, может быть вычислена:

```
str='2+3';
display (str);
%вычисление значения строки
a=eval(str);
display (a);
%вычисление значения строки
a=eval('2*sin(1)');
display (a);
>> stroka
a =
   1.6829
```

Синтаксис следующий:

```
eval (expression)
[output1, ..., outputN] = eval (expression)
```

Функция **eval (expression)** оценивает выражение expression.

Функция **[output1, ..., outputN] = eval (expression)** возвращает вычисленные значения строкового выражения expression в заданные переменные output1, ..., outputN. Тип данных может быть любым.

Ниже использование функции **eval** возвращает четыре матрицы, представляющие магические квадраты чисел от 1 до 4:

```
for n=1:4
 eval('magic(n)');
end
>> plot_tg
ans =
    1
ans =
    1    3
    4    2
ans =
    8    1    6
    3    5    7
    4    9    2
```

```
ans =
   16    2    3   13
    5   11   10    8
    9    7    6   12
    4   14   15    1
```

Код ниже строит символьное выражение. Выходные параметры в виде магической матрицы с размерами 4 × 4 возвращаются в переменную *Z* (рис. 2.6):

```
Z = eval('magic(5)');
figure
mesh(Z)
```

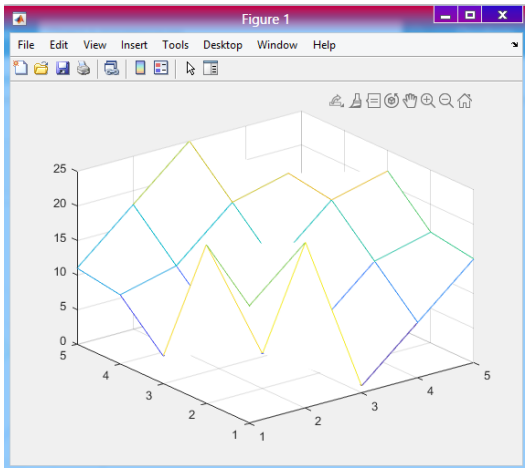


Рис. 2.6. Визуализация символьного выражения функцией **eval**

Функция **evalin (workspace, expression)** используется, чтобы получить значение переменной expression в базовом рабочем пространстве MATLAB и сохранить его в новой переменной. Например, получим значение переменной var в базовом рабочем пространстве MATLAB и сохраним это значение в переменной v:

```
var = magic(5);
v = evalin('base','var')
v =
   17   24    1    8   15
   23    5    7   14   16
    4    6   13   20   22
   10   12   19   21    3
   11   18   25    2    9
```

Команда **T = evalc (S)** выполняет то же, что и функция **eval (s)**, но то, что выводится в командное окно, записывается также и в массив *T*.

2.10.4. Выделение и обработка числовых данных из строки

В процессе анализа данных очень часто приходится выводить данные определенного типа из общего массива данных. Типовая задача – выделение и сохранение в массиве числовых данных из строк. В примере ниже из строки mystring выделяются числа, которые затем сохраняются в вещественном массиве array. Вычисляется сумма элементов массива.

```
mystring = 'sdfkdsgoeskjgk elkr jtk34s ;3k54352642 643l j3kf p35j535';
digits   = regexp(mystring, '[0-9]');
 % array of doubles
array    = double(mystring(digits)')-48;
disp(array);
% sum array of doubles
s=sum(array);
display (s);
Untitled
3    4    3    5    4    3    5    2    6    4    2    6    4    3    3    3    5
5    3    5
s =
   78
```

Вопросы для самоконтроля

1. В чем отличие представления текста в виде массива символов и в виде строковых массивов?
2. Перечислите основные функции для работы со строками. Приведите примеры.
3. Когда необходимы функции преобразования систем счисления? Приведите примеры.
4. Как можно вычислить значение символьной строки? Приведите примеры.
5. Как можно выделить числовые данные из строки? Приведите примеры.

Практическая работа

РАБОТА СО СТРОКАМИ В MATLAB

Задание. Напишите программу. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий содержанию задания;

○ результат работы программы.

Варианты заданий

Вариант 1. Задайте строку, содержащую фамилии, имена обучаемых и их оценки. Напишите программу, преобразующую символьную строку в массив структур. Выведите массив, содержащий отсортированный список фамилий, имен и оценок.

Вариант 2. Задайте строку, содержащую произвольный русский текст длиной не более 200 символов. Проанализируйте строку. Выведите количество вхождений для каждого символа в этой строке. Ответ должен приводиться в грамматически правильной форме, например «а – 18 раз», «ф – 23 раза».

Вариант 3. Задайте строку, содержащую произвольный русский текст длиной не более 40 символов. Осуществите предобработку строки. Удалите из нее все рядом стоящие одинаковые символы, оставив по одному: ПППППОООгоДДДДАААА →ПОГОДА.

Вариант 4. Задайте строку, содержащую произвольный русский текст длиной не более 200 символов. Проанализируйте строку. Выведите символы строки, входящие в нее не более одного раза. Ответ должен приводиться в грамматически правильной форме, например «а б к ь».

Вариант 5. Известны марки машин, изготовляемые в данной стране и импортируемые за рубеж. Дано *N* стран. Проанализируйте данные. Определите для каждой из марок, какие из них были: доставлены во все *N* стран; доставлены в некоторые из *N* стран; не доставлены ни в одну страну.

Вариант 6. Дана строка «Сергей Задорнов 1988 5 4 4 5 3 5». Замените все цифры числительными, например: 2 – «второй», 3 – «третий» и т. д.

Вариант 7. Дана строка, содержащая текст и числа, разделенные пробелами. Выделите из строки числа и сохраните их в числовом массиве. Проанализируйте числовые данные, определите: максимальное и минимальное значения; число, наиболее близкое к минимальному (максимальному) значению.

Вариант 8. Дана строка, содержащая повторяющиеся слова и числа, разделенные пробелами. Например, «аааа 56 в аа ааа сс 34». Выделите из строки уникальные слова и сохраните их в алфавитном порядке в строковом массиве.

Вариант 9. Дана строка, содержащая названия стран, разделенные точками. Названия стран между двумя соседними точками сохраните как имена полей таблицы.

Вариант 10. Дан список хештегов, некоторые из них могут повторяться. Напишите программу, удаляющую повторения хештегов. Результатом должна быть строка хештегов, разделенных пробелом и запятой; приведенных к нижнему регистру. Список может быть пустым. В этом случае должна вернуться пустая строка.

Примечание: хештегом называется непустая строка, состоящая из одного слова.

Вариант 11. Дан список контактов в формате «имя, номер». Напишите программу управления контактами. Для управления необходимо реализовать добавление контакта к существующему списку.

Вариант 12. Дан список контактов в формате «имя, номер». Напишите программу управления контактами. Для управления необходимо реализовать удаление контакта из существующего списка.

Вариант 13. Дан список контактов в формате «имя, номер». Напишите программу управления контактами. Для управления необходимо реализовать поиск контакта в списке по ключу (имени) и вывод найденного контакта.

Вариант 14. Дан список контактов в формате «имя, номер». Напишите программу управления контактами. Для управления необходимо реализовать упорядочивание контактов в алфавитном порядке.

Вариант 15. Напишите программу, преобразующую два своих целочисленных аргумента – числитель m и знаменатель n правильной дроби ($m < n < 100$) – в строку, представляющую собой запись конечной десятичной дроби.

Вариант 16. Напишите программу, преобразующую два своих целочисленных аргумента – числитель m и знаменатель n правильной дроби ($m < n < 100$) – в строку, представляющую собой запись десятичной периодической дроби. Период следует заключить в круглые скобки.

Примечание: значения m, n задайте таким образом, чтобы десятичная дробь была периодической.

Глава 3
ПРОГРАММНЫЕ СРЕДСТВА МАТЕМАТИЧЕСКИХ ВЫЧИСЛЕНИЙ

Система MATLAB – программная среда, ориентированная на математические вычисления. В главе описаны наиболее важные программные средства для выполнения в этой среде массовых математических расчетов самого общего характера. Описание сопровождается множеством примеров, которые рекомендуется повторить, используя командный режим работы или m-файлы примеров. Все описанные операции могут использоваться в составе программ на языке программирования системы MATLAB версии R2020a и выше.

В пособии рассмотрены некоторые функции. Более подробно с описанием и примерами использования всех функций можно познакомиться по ссылке https://docs.exponenta.ru/, набрав в строке поиска название или обозначение функции.

3.1. Базовые математические функции

В системе MATLAB имеется обширная библиотека математических функций. Каждой функции соответствует определенное имя. Функция ставит в соответствие значениям своих аргументов значение результата. Аргументы функции всегда указываются в круглых скобках после имени функции и, если их больше одного, разделяются запятыми. В качестве аргументов могут использоваться другие функции и любые выражения языка MATLAB (при условии соответствия типов аргументов).

Элементарная математическая функция – это, как правило, функция от одной переменной, и в этом случае устанавливается соответствие между массивами значений аргумента и результата.

Аргумент указывается в круглых скобках после имени функции. Имя переменной, которой присваивается значение функции, располагается слева от знака равенства. Если имя присваиваемой переменной не указано, значение функции присваивается служебной переменной ans.

Тип результата вычисления математической функции всегда совпадает с типом ее аргумента. Например, если аргумент функции – вектор-столбец, то значением этой функции также будет вектор-столбец.

Рассмотрим встроенные математические функции системы MATLAB, которые применяются к числам, скалярным переменным и массивам (поэлементно).

Функция **abs** возвращает абсолютное значение. Синтаксис следующий:

```
Y = abs (X)
```

Для массива действительных чисел X команда **Y = abs (X)** возвращает массив *Y* абсолютных значений элементов *X*.

Для массива комплексных чисел *Z* команда **Y = abs (Z)** возвращает массив *Y* модулей комплексных элементов *Z*.

Для строковой переменной *S* команда **Y = abs (S)** возвращает вместо символов, включая пробелы, их ASCII-коды. Например:

```
abs(-5)
ans =     5
abs(3 + 4i)
ans =     5
 ascii = abs('3 + 4I')
 ascii =    51   32   43   32   52   73
 setstr(ascii)
 ans =   3 + 4I
```

Функция **sign** – вычисление знака числа. Синтаксис следующий:

```
S = sign (Z)
```

Для массивов действительных чисел *X* команда **S = sign (X)** возвращает массив *S* тех же размеров, в котором на месте положительного числа стоит единица, на месте нулевого – нуль, на месте отрицательного – (–1).

Для массивов комплексных чисел *Z* команда **S = sign (Z)** возвращает массив комплексных чисел, модуль которых равен единице.

Функции **ceil, fix, floor, round** – округление числа. Синтаксис следующий:

```
Y = ceil (X)
Y = fix (X)
```

Y = floor (X)
Y = round (X)

Для массивов действительных чисел *X*:

– команда **Y = ceil (X)** возвращает значения, округленные до ближайшего целого $\geq X$;

– команда **Y = fix (X)** возвращает значения с усечением дробной части числа;

– команда **Y = floor (X)** возвращает значения, округленные до ближайшего целого $\leq X$;

– команда **Y = round (X)** возвращает значения, округленные до ближайшего целого.

Например, задан одномерный массив действительных чисел *x* = [–1,9 –0,2 3,4 5,6 7,0].

Округлим элементы массива с использованием разных функций. Получим следующие результаты:

ceil(x) ans = –1 0 4 6 7 fix(x) ans = –1 0 3 5 7

floor(x) ans = –2 –1 3 5 7 round(x) ans = –2 0 3 6 7

Для массивов комплексных чисел *Z* эти функции применяются одновременно к действительной и мнимой частям. Например:

```
Z = 2.8+3.5i;
ceil(Z)
floor(Z)
fix(Z)
round(Z)
ans =
   3.0000 + 4.0000i
ans =
   2.0000 + 3.0000i
ans =
   2.0000 + 3.0000i
ans =
      3.00      .0000i
```

Функция **rem** – остаток от деления. Синтаксис следующий:

rem (x, y)

Для действительных чисел *x* и *y* функция **rem (x, y)** вычисляет остаток от деления *x* на *y* или, в других обозначениях, функцию $x(\mathrm{mod}\ y) = x - y \cdot n$, где $n = \mathrm{fix}\ (x/y)$ – ближайшее целое. Для массивов чисел эта функция применяется поэлементно.

ВАЖНО! Входные элементы должны быть целыми числами!

Например, вычислим остаток от деления числа 26 на 5:

```
x=26;
y=5;
z=rem (x, y);
z
z =
  1
```

Вычислим остатки от деления на 3 элементов массива:

```
x=[26 3 7 9];
y=3;
z=rem (x, y);
z
z =
1      0     1     0
```

Функция **gcd** – наибольший общий делитель. Синтаксис следующий:

```
g = gcd (m, n)
[g, c, d] = gcd (m, n)
```

Команда **g = gcd (m, n)** вычисляет наибольший общий делитель двух целых чисел *m* и *n*. Принято, что gcd (0, 0) = 0.

Команда **[g, c, d] = gcd (m, n)**, кроме наибольшего общего делителя, вычисляет два множителя *c* и *d* таких, что выполняется соотношение $g = m \cdot c + n \cdot d$.

Алгоритм:

```
if round (a) ~= a | round (b) ~= b
   error (‘Входные аргументы должны быть целыми числами.')
end
u = [1 0 abs (a)];
v = [0 1 abs (b)];
while v(3)
   q = floor ( u(3) / v(3) );
   t = u - v*q;
   u = v;
   v = t;
end
```

```
    c = u(1) * sign(a);
    d = u(2) * sign(b);
    g = u(3);
```

Например, вычислим НОД чисел 75 и 24:

```
x=75;
y=24;
z=gcd (x,y);
z
z =
    3
```

Для чисел 45 и 36 вычислим НОД и сомножители *c* и *d* (45·1+36 (–1)=9):

```
[g, c, d] = gcd (45, 36);
[g c d] ans = 9     1    -1
```

Для массивов чисел *A* и *B* вычислим сомножители соотношения $g = m \cdot c + n \cdot d$:

```
A = [-5 17; 10 0];
B = [-15 3; 100 0];
G = gcd(A,B)
G = 2×2
   10     1
   10     0
```

Функция **lcm** – наименьшее общее кратное. Синтаксис следующий:

```
g = lcm (m, n)
```

Команда **g = lcm (m, n)** вычисляет наименьшее общее кратное двух целых чисел *m* и *n*. Для массивов чисел эту функцию применять нельзя.

Алгоритм:

```
if round (a) ~= a | round (b) ~= b | a < 1 | b < 1
error (‘Входные аргументы должны быть целыми числами.')
end
c = a*b/gcd (a, b);
```

Например:

```
g = lcm (45, 36)
g =
  180
```

Функции **rat**, **rats** – представление результата в виде рационального числа или цепной дроби. Синтаксис следующий:

[N, D] = rat (X)	rat (X)	S = rats (X)
[N, D] = rat (X, tol)	rat (X, tol)	S = rats (X, tol)

Несмотря на то что все числа с плавающей точкой представлены в компьютере в виде рациональных чисел, иногда целесообразно записать число в виде отношения двух относительно небольших целых чисел. Такое представление на основе цепных дробей и реализуется с использованием вышеперечисленных функций.

Команда **[N, D] = rat (X)** определяет для входа *x* два таких целых числа *n* и *d*, при которых выполняется условие n/d - x <= 1e-6*abs(x).

Команда **[N, D] = rat (X, tol)** позволяет указать точность приближения tol, отличную от значения '1e-6'.

Функции **rat (X)** и **rat (X, tol)** позволяют вывести на экран результат в виде цепной дроби.

Если в качестве входа задан массив чисел *X*, то результатом операций будут массивы соответствующего размера.

Команда **S = rats (X, k)** использует функцию **rat (X)**, чтобы вывести на экран результат в виде простой дроби, точность аппроксимации для которой составляет tol = $10^{(-\mathrm{fix}(k/2))} \cdot \mathrm{abs}(x)$.

Для команды **S = rats (X)** точность аппроксимации принимается по умолчанию равной 1e-6*abs(x), что соответствует значению $k = 13$.

Функция **format rat** равносильна функции **rats**.

Рассмотрим аппроксимацию числа *p* в виде цепной дроби и рационального числа:

```
rat (pi)
ans = 3 + 1/(7 + 1/(16))
rat(pi, 1e-12)
ans = 3 + 1/(7 + 1/(16 + 1/(-294 + 1/(3 + 1/(-4 + 1/(5))))))
[n,d]=rat (pi);
[n d]
ans =   355     113
[n, d]=rat (pi, 1e-12);
[n d]
```

```
ans =    5419351    1725033
s = rats(pi)
s = 355/113
s = rats(pi, 26)
s = 5419351/1725033
```

3.2. Функции для работы с комплексными числами

В MATLAB предусмотрены специальные функции для работы с комплексными числами.

abs	Абсолютное значение и комплексная амплитуда
angle	Аргумент комплексного числа
complex	Создает массив комплексных чисел
conj	Сопряженное комплексное число
cplxpair	Сортировка комплексных чисел в комплексно-сопряженные пары
i	Мнимая единица
imag	Мнимая часть комплексного числа
isreal	Определяет, использует ли массив комплексное устройство хранения данных
real	Действительная часть комплексного числа
sign	Знаковая функция (сигнум-функция)
unwrap	Сдвигает углы фазы

Рассмотрим на примерах, как работают некоторые из перечисленных функций.

Функция **angle** – аргумент комплексного числа. Синтаксис следующий:

```
P = angle (Z)
```

Для массивов комплексных чисел *Z* команда **P = abs (Z)** возвращает массив значений аргументов для элементов *Z*. Значение аргумента измеряется в радианах и находится в пределах от $-p$ до p. Например, для комплексного числа $z = x + iy = re^{ij}$ его модуль r и аргумент j вычисляются следующим образом:

```
r = abs (z)
phi = angle (z)
```

Обратное преобразование выполняет команда

z = r .*exp(i*phi)

Алгоритм. Для вычисления аргумента комплексного числа используется следующее соотношение:

angle(z) = atan2 (imag(z), real(z))

Функции **real, imag** – действительная и мнимая части комплексного числа. Синтаксис следующий:

X = real (Z)

Y = imag (Z)

Для массивов комплексных чисел *Z* команда **X = real (Z)** возвращает массив действительных, а команда **Y = image (Z)** – массив мнимых частей элементов Z.

```
s=3+4i;
X = real (s);
Y = imag (s);
%вывод действительной и мнимой частей
disp (X)
disp (Y)
```

Функция **conj** – операция комплексного сопряжения. Синтаксис следующий:

V = conj (Z)

Для массивов комплексных чисел *Z* команда **V = conj(Z)** возвращает массив комплексно-сопряженных значений для элементов *Z*. Под *комплексно-сопряженными числами* понимается пара комплексных чисел, обладающих одинаковыми действительными частями и равными по абсолютной величине, противоположными по знаку мнимыми частями. Например:

```
Z = 2+3i;
Z
Zc = conj(Z);
Zc
Z =
  2.0000 + 3.0000i
Zc =
  2.0000 - 3.0000i
```

3.3. Трансцендентные функции

Трансцендентные функции – это аналитические функции, не являющиеся алгебраическими.

Алгебраические функции – это функции, которые состоят из цифр и букв, соединяющихся друг с другом при помощи знаков сложения, вычитания, умножения, деления, извлечения корня и возведения в целую степень. Например, $y = x^2 - 3$.

Трансцендентные функции MATLAB представлены ниже.

exp	Показательная функция
ln	Натуральный логарифм
log	Логарифм с произвольным основанием
log10	Десятичный логарифм
log2	Логарифм с основанием 2
^	Возведение выражения в степень
sqrt	Функция квадратного корня
nthroot	Корень *n*-й степени

Рассмотрим некоторые из перечисленных функций.

Функция **sqrt** вычисляет квадратный корень из объекта с помощью алгоритма двоичного поиска. Синтаксис следующий:

c = sqrt (a)
c = sqrt (a, T)
c = sqrt (a, F)
c = sqrt (a, T, F)

Если тип объекта a не задан в качестве входного аргумента, вычисления проводятся согласно следующему внутреннему правилу:

$\text{sign}_c = \text{sign}_a$

$WLc = \text{ceil}\left(\frac{WLa}{2}\right)$

$FLc = WLc - \text{ceil}\left(\frac{WLa - FLa}{2}\right)$,

где WLa, FLa – нижняя и верхняя границы соответственно.

Если числовой тип объекта a – входной аргумент функции, вычисления проводятся согласно правилу распространения типа данных (см. таблицу).

Правила распространения типов данных

Тип входных данных – объект *a*	Тип данных numerictype объекта *T*	Тип выходных данных – объект *a*
Встроенный double	Любой	Встроенный double
Встроенный single	Любой	Встроенный single
fi Fixed	fi Fixed	Тип данных numerictype объекта *T*
fi ScaledDouble	fi Fixed	ScaledDouble со свойствами numerictype объекта *T*
fi double	fi Fixed	fi double
fi single	fi Fixed	fi single
Любой тип данных с фиксированной точкой	fi double	fi double
Любой тип данных с фиксированной точкой	fi single	fi single

Примечание. fi Fixed – это данные с фиксированной точкой.

Команда **c = sqrt (a)** возвращает квадратный корень из объекта *a*. Примеры программ, вычисляющих корень квадратный соответственно из целого, комплексного и отрицательного чисел, приведены ниже:

```
x=sqrt (2);
disp (x);
>> trunc_fun
   1.4142
x=sqrt (2i);
disp (x);
>> trunc_fun
  1.0000 + 1.0000i
x=sqrt (-2);
disp (x);
>> trunc_fun
  0.0000 + 1.4142i
```

Функцию можно применять и к массивам. Продемонстрируем это на примере:

```
a=[-2 -1 0 1 2];
x = sqrt (a);
```

```
disp (x);
>> trunc_fun
  0.0000 + 1.4142i   0.0000 + 1.0000i   0.0000 + 0.0000i   1.0000 +
0.0000i   1.4142 + 0.0000i
```

Так как среди элементов массива есть отрицательные числа, автоматически определен тип «комплексное число» для всех элементов массива.

Если числа положительные, автоматически применяется тип double:

```
a=[5 6 0 1 2];
x = sqrt (a);
disp (x);
 >> trunc_fun
   2.2361   2.4495        0   1.0000   1.4142
```

Команда **c = sqrt (a, T)** возвращает квадратный корень из объекта *a* с типом данных numerictype объекта типа *T* согласно правилу распространения типа данных.

Команда **c = sqrt (a, F)** возвращает квадратный корень из fi-объекта *a*. Numerictype объекта определяется автоматически согласно внутреннему правилу. Когда объект *a* – встроенный тип данных (double или single), этот синтаксис эквивалентен команде **c = sqrt (a)**.

Команда **c = sqrt (a, T, F)** возвращает квадратный корень fi-объекта *a* с типом данных numerictype объекта *T* согласно правилу распространения типа данных. Под fi-объектом в описании понимается тип данных с фиксированной точкой.

Пример

```
w = sqrt((-2:2)')
>> trunc_fun
0.0000 + 1.4142i
0.0000 + 1.0000i
0.0000 + 0.0000i
1.0000 + 0.0000i
1.4142 + 0.0000i
```

Функция **nthroot** – действительный *n*-й корень вещественных чисел. Синтаксис следующий:

```
Y = nthroot (X, N)
```

Команда **Y = nthroot (X, N)** возвращает действительный *n*-й корень элемента *X*. *X* и *N* должны быть действительными скалярами или массивами одного размера. Если элемент *X* отрицательный, соответствующий элемент *N* должен быть нечетным целым числом.

Пример вычисления с положительным вещественным числом:

```
w = nthroot (8, 3);
disp(w);
```

```
>> trunc_fun
    2
```

Пример вычисления с отрицательным вещественным числом:

```
w = nthroot (-8, 3);
disp(w);
```

```
>> trunc_fun
   -2
```

В случае, если *x* – массив, корень *n*-й степени последовательно вычисляется из элементов массива:

```
x=[1 2 4];
w = nthroot (x, 2);
disp(w);
```

```
>> trunc_fun
    1.0000    1.4142    2.0000
```

В качестве массива можно задать несколько действительных корней. Тогда из числа *x* будут последовательно вычисляться степени. Например:

```
N=[1 2 4];
w = nthroot (8, N);
disp(w);
>> trunc_fun
            8.0000    2.8284    1.6818
```

Если *x* – комплексное число, для вычисления *n*-й степени используется функция ^.

```
w = (-27)^(1/3);
disp(w);
>> trunc_fun
    1.5000 + 2.5981i
```

3.4. Тригонометрические функции

Основные тригонометрические и обратные им функции представлены ниже.

sin	Символьная функция синуса
sinc	Нормированная функция синуса
cos	Символьная функция косинуса
tan	Символьная функция тангенса
cot	Символьная функция котангенса
sec	Символьная секущая функция
csc	Символьная функция косеканса
asin	Символьная функция обратного синуса
acos	Символьная функция обратного косинуса
atan	Символьная функция обратного тангенса
acot	Символьная функция обратного котангенса
asec	Символьная обратная секущая функция
acsc	Символьная функция обратного косеканса

Покажем разницу между символьной, нормированной и обратной функциями на примере функции синуса.

Синтаксис функции **sin** следующий:

```
sin (x)
```

Она возвращает значение синуса для числового или символьного аргумента *x*. Например:

```
A = sin([-2, -pi, pi/6, 5*pi/7, 11]);
disp (A);
>> trunc_fun
  -0.9093   -0.0000    0.5000    0.7818   -1.0000
```

Синтаксис функции **sinc** следующий:

```
sinc (x)
```

Она возвращает sin $\pi x/\pi x$, т. е. реализует только символьные результаты. Например:

```
x=pi/2;
a=sinc(x);
disp (a);
>> trunc_fun
  -0.1977
```

Синтаксис функции **asin** следующий:

```
asin (x)
```

Она возвращает функцию обратного синуса от аргумента *X*. Все углы исчисляются в радианах. Для действительных значений *X* в интервале [–1; 1] функция **asin (X)** возвращает значения в интервале [–π/2; π/2]. Для действительных и комплексных значений *X*, не принадлежащих интервалу [–1; 1], функция **asin (X)** возвращает комплексные числа с действительными частями в интервале [–π/2; π/2]. Например:

```
x=0.5;
a=asin(x);
disp (a);
>> trunc_fun
   0.5236
```

Специфичные функции преобразования системы координат и специальные функции, например функции Бесселя, эллиптические функции Якоби и тому подобные, в рамках пособия не рассматриваются.

Вопросы для самоконтроля

1. Перечислите основные математические функции в MATLAB. Приведите примеры их использования.
2. Перечислите основные базовые функции в MATLAB. Приведите примеры их использования.
3. Перечислите основные трансцендентные функции в MATLAB. Приведите примеры их использования.
4. В чем отличие трансцендентных функций от математических и базовых?
5. Перечислите основные тригонометрические функции в MATLAB. Приведите примеры их использования.

Практическая работа

ПРОГРАММНЫЕ СРЕДСТВА МАТЕМАТИЧЕСКИХ ВЫЧИСЛЕНИЙ

Задание. Напишите программы решения задач. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий выполненным расчетам;

○ тестовые данные, на которых проверялась правильность работы программы.

Варианты заданий

Вариант 1. Коэффициенты квадратного уравнения заданы числами *a*, *b*, *c*. Введите пять групп коэффициентов *a*, *b*, *c*. Проанализируйте введенные группы коэффициентов и распределите их по трем категориям: «два корня», «один корень» и «нет корней». Данные сохраните в категориальном массиве.

Вариант 2. Напишите программу, вычисляющую значение выражения $u = \frac{\max^2(x, y, z) - 2^x \cdot \min(x, y, z)}{\sin 2x + \max(x, y, z)/\min(x, y, z)}$ для различных значений группы чисел *x*, *y*, *z*. Проанализируйте введенные значения и распределите их по двум категориям: «действительное значение выражения существует», «действительное значение выражения не существует». Данные сохраните в категориальном массиве.

Вариант 3. Напишите программу вычисления значения функции $x = \frac{\frac{2}{a^2 + 25} + b}{\sqrt{b} + \frac{a + b}{2}}$ для заданной группы коэффициентов *a*, *b*. Проанализируйте введенные значения и распределите их по двум категориям: «действительное значение функции существует», «действительное значение функции не существует». Данные сохраните в категориальном массиве.

Вариант 4. Напишите программу для вычисления значения функции $F(x) = \begin{cases} x^2 - 3x + 9, \text{ если } x \le 3, \\ \frac{1}{x^3 + 6}, \text{ если } x > 3. \end{cases}$

Проанализируйте введенные значения и распределите их по двум категориям: «значение функции при $x \le 3$», «значение функции при $x > 3$». Данные сохраните в категориальном массиве.

Вариант 5. Для различных коэффициентов *a*, *b* напишите программу вычисления значения функции $y = \frac{|a| + 2\sin b}{5{,}5a}$. Проанализируйте введенные значения и распределите их по двум категориям: «действительное значение функции существует», «действительное значение функции не существует». Данные сохраните в категориальном массиве.

Вариант 6. Для различных коэффициентов *x*, *y* напишите программу вычисления значения функции $z = \frac{x + \frac{2+y}{x^2}}{y + \frac{1}{\sqrt{x^2}+10}}$. Проанализируйте введенные значения и распределите их по двум категориям: «действительное значение функции существует», «действительное значение функции не существует». Данные сохраните в категориальном массиве.

Вариант 7. Для различных значений *a*, *b* и *x* напишите программу вычисления значения функции $y = 12ax - \frac{3b\sqrt{6-3a\sin x}}{1+0{,}5\cos bx}$. Проанализируйте введенные значения и распределите их по двум категориям: «действительное значение функции существует», «действительное значение функции не существует». Данные сохраните в категориальном массиве.

Вариант 8. Для различных значений числа *h* напишите программу вычисления значения функции $a = \sqrt{\frac{|\sin 8h| + 17}{(1 - \sin 4h \cdot \cos(h^2 + 18))^2}}$. Проанализируйте введенные значения и распределите их по двум категориям: «действительное значение функции существует», «действительное значение функции не существует». Данные сохраните в категориальном массиве.

Вариант 9. Для различных значений *a* и *h* напишите программу вычисления значения функции $b = 1 - \sqrt{\frac{3}{3 + |\mathrm{tg}ah^2 - \sin ah|}}$. Проанализируйте введенные значения и распределите их по двум категориям: «действительное значение функции существует», «действительное значение функции не существует». Данные сохраните в категориальном массиве.

Вариант 10. Для различных значений *a*, *b* и *x* напишите программу вычисления значения функции $y = \frac{5\sin^2(2x^3)}{2a - x} + 12{,}5\sqrt{b + 2x}$. Проанализируйте введенные значения и распределите их по двум категориям: «действительное значение функции существует», «действительное значение функции не существует». Данные сохраните в категориальном массиве.

Вариант 11. Вычислите для $x = 2$ и $x = 3$ сумму $1 - \frac{2}{3}x + \frac{3}{4}x^2 - \frac{4}{5}x^3 + \ldots + \frac{11}{12}x^{10}$. Проанализируйте введенные значения и распределите их по двум категориям: «сумма отрицательных слагаемых», «сумма положительных слагаемых». Данные сохраните в категориальном массиве.

Вариант 12. Вычислите для $x = 2$ и $x = 3$ сумму $x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots + \frac{x^{11}}{11}$. Проанализируйте введенные значения и распределите их по двум категориям: «значение суммы для $x = 2$», «значение суммы для $x = 3$». Данные сохраните в категориальном массиве.

Вариант 13. Для четного и нечетного значения n вычислите сумму $1 - \frac{1}{2} + \frac{1}{3} - \ldots + (-1)^{n+1}$. Проанализируйте введенные значения и распределите их по двум категориям: «четное n», «нечетное n». Данные сохраните в категориальном массиве.

Вариант 14. Напишите программу для нахождения всех натуральных корней уравнения $x^2 + y^2 = k^2$, где x, y, k лежат в интервалах [1; 10] и [10; 20]. Проанализируйте введенные значения и распределите их по категориям, соответствующим указанным интервалам. Данные сохраните в категориальном массиве.

Вариант 15. Даны натуральные числа m и n. Получите все натуральные числа, меньшие n. Проанализируйте полученные значения и распределите их по двум категориям. К первой отнесите числа, квадрат суммы цифр которых равен m. Ко второй категории отнесите числа, не обладающие таким свойством. Данные сохраните в категориальном массиве.

Вариант 16. Напишите программу нахождения цифрового корня натурального числа. Цифровой корень получается следующим образом. Складываем все цифры этого числа, затем все цифры найденной суммы. Повторяем процесс до тех пор, пока в результате не будет получено однозначное число (цифра), которое и называют цифровым корнем числа.

Глава 4
МАССИВЫ И МАТРИЦЫ

4.1. Понятие массива. Основные функции для работы с массивами

Матрицы и массивы – основное представление информации и данных в MATLAB. По умолчанию предполагается, что каждая заданная переменная – это вектор, матрица или массив. Все определяется конкретным значением переменной. Например, если задано $X = 1$, то это значит, что X – это вектор с единственным элементом, имеющим значение «1», а точнее, матрица с размерами 1×1. Если надо задать вектор из трех элементов, то их значения следует перечислить в квадратных скобках, разделяя пробелами. Например:

```
v=[1 2 3];
disp(v);
```

```
>> vector1
     1     2     3
```

Для вектора-столбца необходимо разделять значения точкой с запятой:

```
v=[1; 2; 3];
disp(v);
```

```
>> vector1
     1
     2
     3
```

Для создания массивов специального вида, например состоящих только из единиц или нулей, в системе MATLAB реализованы специальные функции.

Задание матрицы требует указания нескольких строк и нескольких столбцов. Для разграничения строк используется символ «;» (точка с запятой). Например:

```
v=[1 4 5; 2 6 9; 39 9 2];
disp(v);
```

```
>> vector1
     1     4     5
     2     6     9
    39     9     2
```

Возможен ввод элементов массива в виде арифметических выражений, содержащих любые доступные системе функции, например:

```
v=[1/2+3 4 5; exp(2) 6 9; sin(1/2) 9 2 ];
disp(v);
```

```
>> vector1
    3.5000    4.0000    5.0000
    7.3891    6.0000    9.0000
    0.4794    9.0000    2.0000
```

Для указания отдельного элемента вектора или матрицы используются выражения вида $V(i)$ или $M(i, j)$. Например, если элементу $M(2, 2)$ надо присвоить значение «10», следует записать так:

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

К основным операциям с массивами относят: создание и объединение массивов; определение размера, формы и порядка следования элементов; реконструкцию, индексацию и создание сеток. Каждая операция включает в себя целую группу функций. В рамках пособия подробно рассмотрим только некоторые функции, описание остальных функций, а также примеры работы с ними подробно приведены, например, в [21].

4.2. Создание и объединение массивов

Основные функции для работы с массивами приведены ниже.

zeros	Возвращает массив, содержащий только нули
ones	Возвращает массив, содержащий только единицы
rand	Возвращает массив, значения которого – равномерно распределенные случайные числа
true	Возвращает массив логических единиц
false	Возвращает массив логических нулей
eye (n)	Возвращает матрицу с единицами на главной диагонали и нулями в других местах
diag	Возвращает диагональную матрицу или диагональные элементы матрицы
blkdiag	Блокирование диагональной матрицы
cat	Конкатенация массивов
horzcat	Конкатенация массивов горизонтально

vertcat	Конкатенация массивов вертикально
repelem	Копирование элементов массива
repmat	Копирование массива

Рассмотрим некоторые из перечисленных функций.

Функция **zeros** создает массив нулей. Синтаксис следующий:

```
X = zeros
X = zeros (n)
X = zeros (sz1, ..., szN)
X = zeros (sz)
X = zeros (___, typename)
```

Команда **X = zeros** возвращает скалярный нуль.

Команда **X = zeros (n)** возвращает матрицу нулей с размерами $n \times n$.

Команда **X = zeros (sz1, ..., szN)** возвращает массив нулей с размерами sz1, ..., szN, где sz1, ..., szN – размер каждой размерности.

Команда **X = zeros (sz)** возвращает массив нулей, где вектор размером sz задает размер массива *X*.

Команда **X = zeros (___, typename)** возвращает массив нулей типа typename.

Рассмотрим несколько примеров работы функции.

```
% скалярный ноль
X=zeros;
disp(X);
>> vector2
0
% матрица размером 5x5, состоящая из нулей
n=5;
X=zeros (n);
disp(X);
>> vector2
   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
%матрица, содержащая четыре массива нулей, каждый из которых размером 2x3.
X = zeros(2,3,4);
```

```
>> vector2
(:,:,1) =
    0    0    0
    0    0    0
(:,:,2) =
    0    0    0
    0    0    0
(:,:,3) =
    0    0    0
    0    0    0
(:,:,4) =
    0    0    0
    0    0    0
```

Единичный массив создается аналогичным образом.

Функция **rand** возвращает равномерно распределенные случайные числа. Синтаксис следующий:

```
X = rand
X = rand (n)
X = rand (sz1, ..., szN)
X = rand (sz)
X = rand (___, typename)
X = rand (___, 'like', p)
X = rand (s, ___)
```

Команда **X = rand** возвращает одно равномерно распределенное случайное вещественное число в интервале (0,1).

Команда **X = rand (n)** генерирует матрицу с размерами $n \times n$ равномерно распределенных случайных вещественных чисел из интервала (0, 1).

Команда **X = rand (sz1, ..., szN)** возвращает массив случайных чисел, где sz1, ..., szN указывают на размер каждой размерности. Например, rand (3, 4) возвращает матрицу с размерами 3×4.

Команда **X = rand (sz)** возвращает массив случайных чисел, где вектор размером sz задает размер *X*. Например, rand([3 4]) возвращает матрицу с размерами 3×4.

Команда **X = rand (___, typename)** возвращает массив случайных чисел типа данных typename. Типом данных typename может быть любой тип 'single' или 'double'. Можно комбинировать с любым синтаксисом из перечисленных выше.

Пример. Создадим вектор X, содержащий пять элементов, значения которых – вещественные случайные числа из интервала (0, 1):

```
X=rand (1,5);
disp (X);
>> vector3
   0.0975    0.2785    0.5469    0.9575    0.9649
```

Пример. Создадим матрицу с размерами 5 × 5 равномерно распределенных случайных вещественных чисел из интервала (0, 1):

```
n=5;
X=rand (n);
disp (X);
>> vector3
   0.1576    0.1419    0.6557    0.7577    0.7060
   0.9706    0.4218    0.0357    0.7431    0.0318
   0.9572    0.9157    0.8491    0.3922    0.2769
   0.4854    0.7922    0.9340    0.6555    0.0462
   0.8003    0.9595    0.6787    0.1712    0.0971
```

Если необходимо сгенерировать матрицу с размерами $M \times N$, содержащую числа из некоторого интервала [A; B], используется формула:

$$X = A + (B - A)\mathrm{rand}(M, N).$$

Пример. Сгенерируем матрицу с размерами 3 × 2, содержащую равномерно распределенные вещественные случайные числа из интервала (–5; 5):

```
X = -5 + (5+5)*rand(3,2);
disp (X);
>> vector3
   3.2346    4.5022
   1.9483   -4.6555
  -1.8290   -0.6126
```

Если необходимо сгенерировать вектор или матрицу, содержащие целые числа из какого-либо диапазона, вместо функции **rand** используется функция **randi**. Например, сгенерируем вектор, содержащий пять случайных целых чисел из диапазона [10; 50]:

```
X = randi([10 50],1,5);
disp (X);
>> vector3
   25    41    42    17    30
```

Функция **diag** возвращает диагональную матрицу. Синтаксис следующий:

```
D = diag (v)
D = diag (v, k)
x = diag (A)
x = diag (A, k)
```

Команда **D = diag (v)** возвращает диагональную матрицу или элементы главной диагонали.

Команда **D = diag (v, k)** возвращает вектор из элементов *k*-й диагонали. Если $k = 0$, то команда возвращает элементы основной диагонали, если $k > 0$, то возвращаются элементы выше основной диагонали, если $k < 0$, то возвращаются элементы ниже основной диагонали.

Команда **x = diag (A)** возвращает основную диагональ матрицы *A*.

Команда **x = diag (A, k)** возвращает элементы *k*-й диагонали матрицы *A*.

Пример. Выведем на экран элементы главной диагонали матрицы:

```
X = [1 4 5;  2 6 9;  39 9 2];
A=diag (X);
disp (A);
>> vector3
    1
    6
    2
```

Пример. Извлечем из матрицы *X* элементы диагонали, расположенные выше главной диагонали:

```
X = [1 4 5;  2 6 9;  39 9 2];
A=diag (X,1);
disp (A);
>> vector3
    4
    9
```

Функция **blkdiag** создает матрицу с диагональю из блоков других матриц меньшего размера. Синтаксис следующий:

```
B = blkdiag (A1, ..., AN)
```

Команда **B = blkdiag (A1, ..., AN)** возвращает блочную диагональную матрицу, созданную путем выравнивания входных матриц *A*1, ..., *AN* по диагонали *B*.

$$\begin{bmatrix} A1 & 0 & 0 & 0 \\ 0 & A2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & AN \end{bmatrix}$$

4.3. Определение размера, формы и порядка массива

Функции, с помощью которых можно определить размер, форму и порядок следования элементов массива, представлены ниже.

length	Длина самого большого измерения массива
size	Размер массивов
ndims	Количество измерений массива
numel	Количество элементов массива
isscalar	Определяет, является ли входной параметр скаляром
issorted	Определяет, отсортирован ли массив
issortedrows	Определяет, отсортированы ли строки матрицы или таблицы
isvector	Определяет, является ли входной параметр вектором
ismatrix	Определяет, является ли входной параметр матрицей
isrow	Определяет, является ли входной параметр вектором-строкой
iscolumn	Определяет, является ли входной параметр вектором-столбцом
isempty	Определяет, пуст ли массив

Рассмотрим некоторые из вышеперечисленных функций.

Функция **length** – длина самого большого измерения в массиве. Синтаксис следующий:

L=length (X)

Функция возвращает длину самого большого измерения в массиве *X*. Для векторов длина – просто число элементов. Для массивов с большим количеством размерностей длиной является наибольшая размерность max(size(X)). Длина пустого массива – нуль.

В качестве примера создадим массив с размерами 2 × 3. Самый большой размер – количество столбцов (а именно 3).

```
X = [1 4 5; 3 7 9];
L=length (X);
```

```
disp (L);
>> vector3
    3
```

Функция **ismatrix** определяет, является ли входной параметр матрицей. Синтаксис следующий:

```
TF=ismatrix (X)
```

Функция возвращает логическую единицу (TRUE), если A – матрица. В противном случае возвращает логический ноль (FALSE). Матрица A – двумерный массив с размерами $M \times N$, где M и N – неотрицательные целые числа.

Если мы используем матрицу из примера выше, то в качестве ответа получим TRUE, так как массив X – матрица.

```
X = [1 4 5; 3 7 9];
TF=ismatrix (X);
disp (TF);
>> vector3
  1
```

4.4. Создание сеток

Сетка в контексте массивов понимается как задание координат для разметки поверхности трехмерного графика. Основные функции, реализующие такую возможность, представлены ниже.

linspace	Создание вектора с линейно распределенными значениями
logspace	Создание вектора с логарифмически распределенными значениями
freqspace	Частотный интервал для частотной характеристики
meshgrid	2D- и 3D-сетки
ndgrid	Прямоугольная сетка в пространстве N – D

Рассмотрим некоторые из перечисленных функций.

Синтаксис функции **linspace** следующий:

```
y = linspace (x1, x2)
y = linspace (x1, x2, n)
```

Команда **y = linspace (x1, x2)** возвращает вектор-строку из 100 равномерно разнесенных точек, принадлежащих интервалу $[x_1; x_2]$.

Команда **y = linspace (x1, x2, n)** возвращает вектор-строку из n равномерно разнесенных точек, принадлежащих интервалу $[x_1; x_2]$.

В качестве примера создадим вектор из семи равномерно разнесенных точек из интервала [–5; 5]:

```
Y = linspace(-5,5,7);
disp (Y);
```

```
>> vector3
   -5.0000   -3.3333   -1.6667         0    1.6667    3.3333    5.0000
```

Синтаксис функции **logspace** следующий:

```
y = logspace (a, b)
y = logspace (a, b, n)
y = logspace (a, pi)
y = logspace (a, pi, n)
```

Команда **y = logspace (d1, d2)** формирует вектор-строку, содержащую 50 равноотстоящих в логарифмическом масштабе точек, которые покрывают диапазон от 10^{d1} до 10^{d2}.

Команда **y = logspace (d1, d2, n)** формирует вектор-строку, содержащую *n* равноотстоящих в логарифмическом масштабе точек, которые покрывают диапазон от 10^{d1} до 10^{d2}.

Команда **y = logspace (a, pi)** генерирует 50 точек на интервале $[10^a; \pi]$, что полезно в цифровой обработке сигналов.

Команда **y = logspace (a, pi, n)** генерирует *n* точек на интервале $[10^a; \pi]$.

В качестве примера создадим вектор из пяти точек, расположенных в интервале $[10^1; 10^5]$:

```
Y = logspace (1, 5, 5);
disp (Y);
```

```
>> vector3
          10         100        1000       10000      100000
```

4.5. Изменение и реконструкция массивов

Функции, изменяющие массив, представлены ниже.

sort	Сортировка массива
sortrows	Сортировка строк массива или таблицы
flip	Инвертирует порядок элементов
fliplr	Отражает массив слева направо
flipud	Отражает массив сверху вниз
rot90	Вращает массив на 90°

transpose	Транспонирует вектор или матрицу
ctranspose	Комплексное сопряженное транспонирование
permute	Перестановка измерений массива
ipermute	Обратное преобразование измерений массива, противоположное permute
circshift	Циклический сдвиг элементов массива
shiftdim	Сдвиг измерений массива
reshape	Изменение размерности массива
squeeze	Удаление размерности, равной единице

Рассмотрим некоторые из перечисленных функций.

Синтаксис функции **flip** следующий:

```
B = flip (A)
B = flip (A, dim)
```

Команда **B = flip (A)** возвращает массив *B* того же размера, что и массив *A*, но с инвертированным порядком следования элементов. Размерность, которая переупорядочивается в массиве *B*, зависит от формы массива *A*:

– если *A* – вектор, функция **flip (A)** инвертирует порядок элементов вдоль вектора;

– если *A* – матрица, функция **flip (A)** инвертирует элементы в каждом столбце;

– если *A* – массив N – D, функция **flip (A)** работает с первой размерностью *A*, значением размера которой не является единица.

В качестве примера рассмотрим, как получить массив *B* – зеркальное отражение массива *A*.

```
A = ‘no word, no bond, row on.’;
B = flip(A);
disp (B);
>> vector3
.no wor ,dnob on ,drow on
```

Создадим диагональную матрицу *A* и получим матрицы *B* и *C*, являющиеся соответственно прямым зеркальным отражением и зеркальным отражением элементов второго измерения, т. е. столбцов матрицы *A*:

```
A = diag([100 200 300]);
disp ('исходная матрица');
disp (A);
```

```
disp ('обратное отражение исходной матрицы');
B=flip (A);
disp (B);
disp ('обратное отражение порядка столбцов исходной матрицы');
C=flip (A,2);
disp ©;
>> vector3
исходная матрица
   100     0     0
     0   200     0
     0     0   300
обратное отражение исходной матрицы
     0     0   300
     0   200     0
   100     0     0
обратное отображение порядка следования столбцов исходной матрицы: первый столбец стал последним, последний – первым и т. д.
     0     0   100
     0   200     0
   300     0     0
```

Синтаксис функции **flipud** следующий:

```
B = flipud (A)
```

Функция возвращает матрицу *A* с обратным порядком следования элементов в столбцах. Например:

```
A = diag([100 200 300]);
disp ('исходная матрица');
disp (A);
D=flipud(A);
disp ('обратное отражение элементов в столбцах исходной матрицы');
disp (D);
>> vector3
исходная матрица
   100     0     0
     0   200     0
     0     0   300
```

обратное отражение элементов в столбцах исходной матрицы

```
     0     0   300
     0   200     0
   100     0     0
```

4.6. Индексация элементов массива

Под индексацией в контексте массивов понимают извлечение из них элементов или блоков. Функции индексации элементов массива представлены ниже.

colon	Создание вектора, индексирование массивов и for-итерация элементов массива
end	Завершает блок кода или указывает на последний индекс массива
ind2sub	Преобразует линейные индексы элементов матрицы в индексы (*i*, *j*)
sub2ind	Преобразует индексы (*i*, *j*) в линейные индексы

Рассмотрим некоторые функции.

Синтаксис функции **colon** следующий:

```
x = j : k
x = j: i: k
A (:, n)
A (m, _)
A (_, m)
A (j: k)
```

Двоеточие – один из операторов в MATLAB. С его помощью можно создать векторы, массивы индексов и задать for-итерации для элементов массива.

Так, команда **x = j: k** создает вектор *x* с элементами, индексы которых принадлежат промежутку [*j*, *j*+1, *j*+2, …, *j*+*m*], где *m* = fix(*k* – *j*). Если *j* и *k* – целые числа, то индексы элементов массива принадлежат промежутку [*j*; *k*].

Например, создадим вектор, элементы которого – целые числа из интервала [1; 10]. Оператор «двоеточие» использует шаг по умолчанию +1:

```
x = 1:10;
disp (x);
```

```
>> vector3
 1    2    3    4    5    6    7    8    9   10
```

Пример случая, когда *j* и *k* – вещественные числа:

```
x = 1.5:10;
disp (x);
>> vector3
  1.5000    2.5000    3.5000    4.5000    5.5000    6.5000    7.5000
  8.5000    9.5000
```

Команда **x = j:i:k** создает вектор *x*, значения индексов элементов которого задаются с шагом *i*.

Если *i* – нецелое число, работает арифметика с плавающей точкой, что играет роль в вопросе включения конечной точки *k* в вектор, так как в этом случае *k* не может быть точно равно $j + m \cdot i$.

A (:, n), A (m, :), A (:), A (j: k) – общие выражения индексации для случая, когда *A* – матрица.

Двоеточие может использоваться в качестве индекса в выражении индексации, таком как **A(:, n)**. Это действует как сокращение, чтобы включать все индексы в конкретное измерение массива. Также можно создать вектор с двоеточием в целях индексации, такой как **A(j:k)**. Некоторые выражения индексации комбинируют оба способа использования двоеточия, как в выражении **A(:, j: k)**.

Общие выражения индексации, которые содержат двоеточие, следующие:

– A (:, n) – *n*-й столбец матрицы *A*;

– A (m, _) – *m*-я строка матрицы *A*;

– A (:, :, p) – *p*-я страница 3D-массива *A*;

– A (_) организует из элементов массива *A* вектор-столбец. Если массив *A* уже им является, изменения не вступают в силу;

– A (:, _) организует из элементов массива *A* двумерную матрицу. Действие не выполняется, если массив *A* уже матрица или вектор;

– A (j: k) преобразует в вектор элементы матрицы *A*. Так, если *j* = 1, а *k* = 3, в вектор организуются элементы первого столбца. Если *j* = 4, *k* = 6, в вектор преобразуются элементы второго столбца и т. д. То есть матрица *A* становится эквивалентна вектору, элементы которого имеют индексы [*A*(*j*), *A*(*j*+1), …, *A*(*k*)];

– A (:, j: k) возвращает матрицу со столбцами [A(:, j), A(:, j+1), …, A(:, k)].

Рассмотрим несколько примеров.

1. Индексируем первую строку:

```
A = [1 2 3; 7 8 9; 11 90 5];
disp («исходная матрица»);
disp (A);
disp («индексация первой строки»);
disp (A(1,_);
 >> vector3
     1    2    3
     7    8    9
    11   90    5
 >> vector3
    1    2    3
```

2. Индексируем второй и третий столбцы:

```
A = [1 2 3; 7 8 9; 11 90 5];
disp («исходная матрица»);
disp (A);
disp («индексация 2 и 3 столбцов»);
disp (A(:,2:3));
>> vector3
исходная матрица
     1    2    3
     7    8    9
    11   90    5
индексация 2 и 3 столбцов
      4    3
     8    9
    90    5
```

3. Преобразуем форму матрицы. Сделаем из матрицы вектор:

```
A = [1 2 3; 7 8 9; 11 90 5];
disp («исходная матрица»);
disp (A);
disp («преобразование в вектор»);
disp (A(_));
>> vector3
исходная матрица
     1    2    3
     7    8    9
    11   90    5
```

преобразование в вектор

1
7
11
2
8
90
3
9
5

Синтаксис функции **ind2sub** следующий:

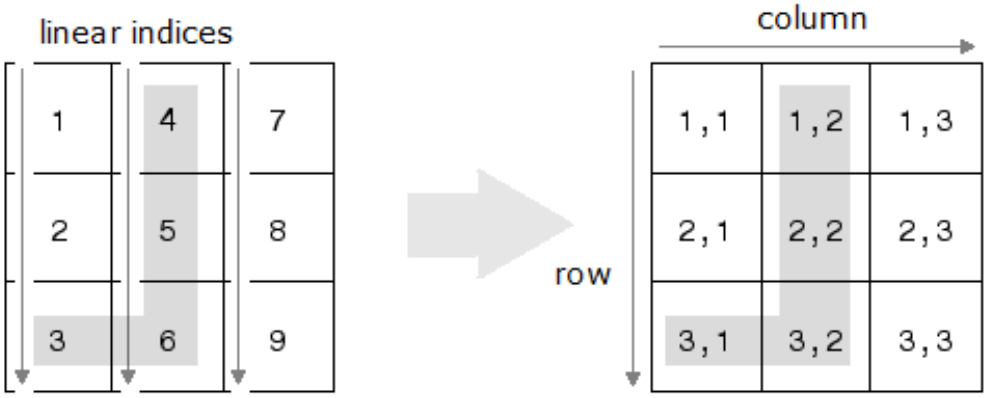
[row, col] = ind2sub (sz, ind)

[I1, I2, …, In] = ind2sub (sz, ind)

Команда **[row, col] = ind2sub (sz, ind)** возвращает массивы row и col, содержащие эквивалентные индексы строки и столбца, соответствующие линейным индексам ind для матрицы с размерами sz. Здесь sz – вектор с двумя элементами: sz(1) задает количество строк, sz(2) – количество столбцов.

Команда **[I1, I2, …, In] = ind2sub (sz, ind)** возвращает *n* массивов I1, I2, …, In, которые содержат эквивалентные многомерные индексы, соответствующие линейным индексам ind для многомерного массива с размерами sz. Здесь sz – вектор с *n* элементами, который задает размер каждого измерения массива.

В качестве примера преобразуем линейные индексы, содержащиеся в матрице *A*, в индексы строки и столбца матрицы *B* с размерами 3 × 3. Процесс преобразования проиллюстрирован на рисунке.



Процесс преобразования индексов

```
ind=[3 4 5 6];
sz=[3 3];
[row, col]=ind2sub (sz, ind);
disp (row);
disp (col);
   >> vector4
     3     1     2     3
     1     2     2     2
```

4.7. Удаление строк или столбцов из матрицы

Самый легкий способ удалить строку или столбец из матрицы – указать для номера строки или столбца пустые квадратные скобки [].

Например, создадим матрицу с размерами 4 × 4 и удалим вторую строку:

```
A = magic(4)
A = 4×4
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
A(2,:) = []
A = 3×4
    16     2     3    13
     9     7     6    12
     4    14    15     1
```

Удалим третий столбец:

```
   A(:,3) = []
   A = 3×3
   16     2    13
    9     7    12
    4    14     1
```

Вопросы для самоконтроля

1. Как создать массив?

2. Чем отличается процесс создания обычного массива от создания массива специального вида? Продемонстрируйте на примерах.

3. Как в контексте массивов понимается сетка? Приведите пример создания сетки.

4. Перечислите основные функции, определяющие размер массива. Приведите примеры.

5. Перечислите основные функции, определяющие форму массива. Приведите примеры.

6. Перечислите основные функции, определяющие порядок элементов массива. Приведите примеры.

7. Перечислите основные функции, изменяющие массив. Приведите примеры.

8. Перечислите основные функции, реконструирующие массив. Приведите примеры.

9. Объясните, почему, в отличие от операций сложения и вычитания, можно перемножать матрицы разной размерности. Какие параметры размерностей перемножаемых матриц должны совпадать, чтобы не произошло ошибки?

10. Объясните, почему операцию «возведение в степень» можно проводить только с квадратными матрицами и целыми степенями?

11. Как удалить строку (или столбец) в массиве?

12. Чем обычный массив отличается от временных рядов, расписания, таблиц, структур, datetime-массивов, массивов ячеек и категориальных массивов?

13. Можно ли преобразовать обычный массив во временной ряд, массив ячеек, таблицу, расписание, структуру, категориальный массив? Как это сделать? Приведите пример. Когда необходимы такие преобразования? Приведите примеры.

14. Какую важную для анализа данных информацию можно получить с помощью специальных функций для работы с массивами? Перечислите их. Приведите примеры использования.

Практическая работа

МАССИВЫ В MATLAB

Задание. Напишите программы решения задач. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий выполненным расчетам;

○ тестовые данные, на которых проверялась правильность работы программы.

Варианты заданий

Вариант 1. В одномерном массиве, состоящем из *n* вещественных чисел, вычислите: а) сумму отрицательных элементов массива; б) произведение элементов массива, расположенных между максимальным и минимальным элементами.

Вариант 2. Дан двумерный массив строк. Найдите номер последней по порядку строки массива, содержащей наибольшее количество букв «ш» и «щ».

Вариант 3. Выполните сжатие одномерного массива, удалив из него все элементы, значения которых меньше заданного числа *n*. Освободившиеся в конце массива места заполните единицами.

Вариант 4. В одномерном массиве, состоящем из *n* вещественных чисел, вычислите: а) сумму положительных элементов массива; б) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Вариант 5. В прямоугольной целочисленной матрице с размерами 3 × 4 определите количество строк, не содержащих ни одного нулевого элемента.

Вариант 6. В одномерном массиве, состоящем из *n* целых чисел, вычислите: а) произведение элементов массива с четными номерами; б) сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Вариант 7. Упорядочьте одномерный массив целых чисел методом быстрой сортировки. Суть алгоритма: массив разделяется на две части относительно среднего элемента. В левую часть помещаются элементы, меньшие среднего элемента, в правую – большие среднего элемента. Это достигается путем просмотра массива попеременно с обоих концов, при этом каждый элемент сравнивается со средним, элементы, находящиеся в «неподходящей» части, меняются местами. Далее процедура разделения повторяется отдельно для левой и правой частей. В каждой части выбирается очередной средний элемент.

Вариант 8. В одномерном массиве, состоящем из *n* вещественных чисел, вычислите: а) сумму элементов массива с нечетными номерами; б) сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Вариант 9. Задан двумерный массив строк. Найдите номер первой по порядку строки массива, содержащей наибольшее число цифр.

Вариант 10. Задан двумерный массив строк. Выведите слова, образованные нечетными элементами каждого столбца массива.

Вариант 11. В одномерном массиве, состоящем из *n* целых чисел, вычислите: а) сумму элементов массива, расположенных правее последнего отрицательного элемента; б) сумму элементов массива, расположенных левее первого отрицательного элемента.

Вариант 12. Задан двумерный массив строк. В каждой строке массива найдите количество букв «е», расположенных справа от буквы «н».

Вариант 13. Напишите программу, которая для прямоугольной целочисленной матрицы определяет номер самого левого столбца, содержащего только положительные элементы. Если такого столбца нет, выведите соответствующее положение.

Вариант 14. Преобразуйте одномерный массив таким образом, чтобы сначала в нем располагались элементы, равные нулю, затем все остальные в порядке их следования в исходном массиве.

Вариант 15. Упорядочьте строки прямоугольной целочисленной матрицы с размерами 3 × 4 по возрастанию сумм их элементов.

Вариант 16. Напишите программу, которая для целочисленной матрицы с размерами 5 × 6 вычисляет среднее арифметическое ее элементов и количество положительных элементов в каждой строке.

Практическая работа

РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ ПРИ ПОМОЩИ МАТРИЦ И ВЕКТОРОВ

Решение систем линейных уравнений (СЛУ) – одна из важных проблем анализа данных.

Рассмотрим один из методов решения СЛУ – метод обратной матрицы.

Пусть дана следующая СЛУ: $\begin{cases} 4a + b - c = 6, \\ a - b + c = 4, \\ 2a - 3b - 3c = 4. \end{cases}$

Перепишем систему в виде матрицы *A* и вектора *B*:

$$A = \begin{vmatrix} 4 & 1 & -1 \\ 1 & -1 & 1 \\ 2 & -3 & -3 \end{vmatrix} \text{ и } B = \begin{vmatrix} 6 \\ 4 \\ 4 \end{vmatrix}$$

Проверим необходимое условие существования решения для СЛУ: определитель матрицы, составленный из коэффициентов левой части уравнений, не должен быть равен нулю. В MATLAB это можно проверить функцией **det ()**.

Если $\det(A) \neq 0$, с помощью функции **inv ()** найдем матрицу, обратную матрице *A*. Решение СЛУ в MATLAB находится как произведение обратной матрицы и вектора свободных членов:

```
x=inv(A)*B
A=[4 1 -1;1 -1 1;2 -3 -3];
B=[6;4;4];
if det (A)~=0
  x=inv(A)*B
  disp(x)
else
  disp('нет решений')
end
2
-1
1
```

Задание. При помощи матриц и векторов решите систему уравнений.

Варианты заданий

Вариант 1.

$$\begin{cases} 1{,}2x_1 + 0{,}3x_2 - 0{,}2x_3 = 1{,}3 \\ 0{,}5x_1 + 2{,}1x_2 + 1{,}3x_3 = 3{,}9 \\ -0{,}9x_1 + 0{,}7x_2 + 5{,}6x_3 = 5{,}4 \end{cases}$$

Вариант 2.

$$\begin{cases} x + y + z = 3 \\ x + 2y + 3z = 6 \\ x + 3y + 6z = 10 \end{cases}$$

Вариант 3.

$$\begin{cases} x + y + z = 4 \\ x + 2y + 3z = 11 \\ x + 3y + 6z = 18 \end{cases}$$

Вариант 4.

$$\begin{cases} x + y + z = 3 \\ x + 2y + 3z = 8 \\ x + 3y + 6z = 13 \end{cases}$$

Вариант 5.

$$\begin{cases} x + y + z = 1 \\ x + 2y + 3z = 1 \\ x + 3y + 6z = 0 \end{cases}$$

Вариант 6.

$$\begin{cases} x + y + z = 5 \\ x + 6y + x = 10 \\ x + y + 6z = 0 \end{cases}$$

Вариант 7.

$$\begin{cases} 2x_1 + 3x_2 + 2x_3 = 9 \\ x_1 + 2x_2 - 3x_3 = 14 \\ 3x_1 + 4x_2 + x_3 = 16 \end{cases}$$

Вариант 8.

$$\begin{cases} x_1 + 2x_2 - 3x_3 = 14 \\ -x_2 + 2x_3 = -7 \\ 3x_1 + 7x_3 = 0 \end{cases}$$

Вариант 9.

$$\begin{cases} x_1 - 2x_2 - 3x_3 = 0 \\ 2x_1 + x_2 + 3x_3 = 1 \\ 3x_1 - x_2 - 2x_3 = 3 \end{cases}$$

Вариант 10.

$$\begin{cases} 2x_1 + 2x_2 + 2x_3 = 4 \\ x_1 + x_2 - x_3 = 0 \\ 3x_1 + 3x_2 - x_3 = 2 \\ -x_1 - x_2 + 3x_3 = 2 \end{cases}$$

Вариант 11.

$$\begin{cases} 3x + 4y - 5z = 1 \\ 7x - 4y + z = -2 \\ 2y - 9z = 5 \end{cases}$$

Вариант 12.

$$\begin{cases} 3x - 3y + 2z = 2 \\ 4x - 5y + 2z = 1 \\ 5x - 6y + 4z = 3 \end{cases}$$

Вариант 13.

$$\begin{cases} 3x + 2y - z = 3 \\ x - y + 2z = -4 \\ 2x + 2y + z = 4 \end{cases}$$

Вариант 14.

$$\begin{cases} 3x + 2y - z = 3 \\ -5y + 7z = -15 \\ z = 0 \end{cases}$$

Вариант 15.

$$\begin{cases} 2x - y - 3z = 3 \\ 3x + 4y - 5z = -8 \\ 2y + 7z = 17 \end{cases}$$

Вариант 16.

$$\begin{cases} x_1 + 2x_2 - x_3 = 4 \\ 3x_1 + 2x_3 = -8 \\ 4x_1 - 2x_2 + 5x_3 = 0 \end{cases}$$

Глава 5
СИМВОЛЬНЫЕ РАСЧЕТЫ В MATLAB. ВОЗМОЖНОСТИ ПАКЕТА SYMBOLIC MATH TOOLBOX

5.1. Создание символьного выражения в MATLAB

В анализе данных символьные выражения используются тогда, когда вместо численных необходимы аналитические вычисления. Ниже приведены способы задания символьных выражений.

sym	Создает символьные переменные, выражения, функции, матрицы
syms	Создает символьные переменные и функции
symfun	Создает символьные функции
str2sym	Оценивает строку, представляющую символьное выражение

Рассмотрим первых три способа.

Использование оператора **syms**:

```
syms x y;
```

Таким простым способом мы создали две символьные переменные. Пока они ничего не делают и не представляют какой-либо ценности, но чуть позже мы увидим, что они могут быть полезны.

Поддержка векторов символов, которые не являются допустимыми именами переменной и не задают номер, в версии R2020a удалена. Чтобы создать символьные выражения, сначала создают символьные переменные, а затем используют операции на них. Например, создадим символьную переменную *x*, которую затем включим в символьное выражение *x* + 1. Создадим две символьные переменные *x* и *y*, которые затем включим в выражение *x* + 2*y*, и вычислим его значение при *x* = 1 и *y* = 2:

```
syms x, y;
f(x, y) = x + 2 * y;
f(1, 2)
ans = 5
```

Использование оператора **sym**:

```
x = sym ('x')
A = sym ('a', [n1 ... nM])
A = sym ('a', n)
sym (___, set)
```

sym (___, 'clear')
sym (num)
sym (num, flag)
symexpr = sym (h)

Команда **X = sym('x')** создает символьную переменную *x*. Например:

```
f = sym('f');
x = sym('x');
y = sym('y');
```

Команда **A = sym ('a', [n1 ... nM])** создает n1 ... nM символьный массив, заполненный автоматически сгенерированными элементами. Например, A = sym ('a', [1 3]) создает вектор-строку A = [a1 a2 a3]. Сгенерированные элементы a1, a2 и a3 не появляются в рабочей области MATLAB. Например:

```
A = sym('a',[1 3]);
disp(A);
a1='ff';
a2='3x+2';
disp (a1)
disp (a2)
categ_array
[a1, a2, a3]
ff
3x+2
```

Можно сразу задать функцию, полином или выражение, описывающие анализируемый процесс (объект или данные):

```
f = sym('cos(x) + cos(y)')
```

Команда **A = sym ('a', n)** создает *n* символьную матрицу, заполненную автоматически сгенерированными элементами, имеющими форму Ai_j:

```
A = sym ('A',[3 4])
A =
[ A1_1, A1_2, A1_3, A1_4]
[ A2_1, A2_2, A2_3, A2_4]
[ A3_1, A3_2, A3_3, A3_4]
```

Символьные выражения в анализе данных полезны тем, что вычисления с ними производятся без погрешностей. Возможны следующие типы преобразований.

simplify	Алгебраическое упрощение
simplifyFraction	Упрощение символьных рациональных выражений
coeffs	Коэффициенты полинома
expand	Раскрытие скобок
numden	Извлекает числитель и знаменатель
partfrac	Осуществляет разложение элементарной дроби
children	Вычисление подвыражений символьного выражения
collect	Вычисление коэффициентов при степенях многочлена
divisors	Вычисление делителей целого числа или выражения

Рассмотрим некоторые из перечисленных функций.

Функция **expand** – раскрытие скобок. Синтаксис следующий:

```
expand (S)
expand (S, Name, Value)
```

Функция **expand (S)** расширяет символьное выражение *S*. Рациональные выражения функция раскладывает на простые дроби, полиномы – на полиномиальные выражения и т. д. Для примера зададим символьное выражение и попробуем раскрыть скобки:

```
syms x y;
f = (x+2)*(x+3) + (y+5);
f = expand(f);
f =
x^2 + 5*x + y + 11
```

Функция **expand (S, Name, Value)** расширяет символьное выражение *S*, при этом использует дополнительные параметры, указанные одним или несколькими аргументами пары Name – Value.

Например, указание true в качестве значения параметра IgnoreAnalyticConstraints позволяет значительно упростить вывод. Как видно из примера ниже, добавление этого параметра позволяет получить многочлен первой степени вместо многочлена второй степени. Это значительно упрощает дальнейшие вычисления и анализ.

```
syms a b c
f = log((a*b/c)^2);
```

```
disp ('упрощение без указания дополнительных параметров');
expand(f)
disp ('упрощение с указанием дополнительного параметра
IgnoreAnalyticConstraints');
expand(f,'IgnoreAnalyticConstraints',true)
упрощение без указания дополнительных параметров
ans =
log((a^2*b^2)/c^2)
упрощение с указанием дополнительного параметра
IgnoreAnalyticConstraints
ans =
2*log(a) + 2*log(b) - 2*log(c)
```

Функция **simplify** также упрощает символьное выражение. Синтаксис следующий:

```
simplify (f)
simplify (f, options)
```

Функция **simplify (f)** выполняет алгебраическое упрощение выражения *f*. Например:

```
syms x y;
f = (x*y^3 + 2*y + x^4*y)/(y);
f = simplify(f)
f =
x^4 + x*y^2 + 2
```

Работает эта функция медленней, чем функция **expand**, но более эффективно и конфигурируемо. Для примера упростим выражение $\sin x^2 + \cos x^2$ двумя способами: с помощью функции **expand** и функции **simplify**:

```
syms x y
f = (sin(x)^2+cos(x)^2);
disp ('упрощение expand');
expand(f)
disp ('упрощение simplify');
simplify(f)
>> function
упрощение expand
ans =
cos(x)^2 + sin(x)^2
```

упрощение simplify
ans =
1

Как видно из примера, упрощение с помощью функции **simplify** гораздо эффективней в сравнении с функцией **expand**.

Функция **simplify (f, options)** выполняет то же, что и функция **simplify (f)**. Результат поиска можно изменить с помощью опций (options). Ниже приведены наиболее часто используемые опции.

All	Возвращает список всех эквивалентных выражений, а не только одного, наиболее простого
Discard=false	Продолжает поиск других эквивалентных выражений после нахождения наиболее простого
IsSimple=false	Возвращает список эквивалентных выражений, которые значительно сложнее, чем наиболее простое

Приведем примеры.

Для выражения $\sin^2(x)+\cos^2(x)$ командой

S = simplify(sin(x)^2 + cos(x)^2)

будет возвращено эквивалентное выражение ‘1’.

Командой **S = simplify(sin(x)^2 + cos(x)^2,'Steps',10,'All',true)** этому же выражению будет возвращен список эквивалентных выражений:

S =
1
cos(x)^2 + sin(x)^2

Функция **factor** – разложение на простые множители. Синтаксис следующий:

f = factor(n)

Команда **f = factor (n)** возвращает вектор-строку, содержащий простые множители числа *n*. Вектор *f* имеет тип данных, совпадающий с числом *n*. Данная функция помогает преобразовать символьное выражение, например, в полином в MATLAB. Это важно, например, для определения вида тренда:

```
syms x;
f = (x+4)^4 + x^2 + (x-2)^3;
f = factor(f)
f =
x^4 + 17*x^3 + 91*x^2 + 268*x + 248
```

5.2. Вычисление значений символьных выражений в MATLAB

Функция **subs** осуществляет символьную замену. Синтаксис следующий:

snew = subs (s, old, new)
snew = subs (s, new)
snew = subs (s)

Команда **snew = subs (s, old, new)** возвращает копию выражения *s*, заменяя все значения аргумента old на значение аргумента new, и затем оценивает *s*. Здесь *s* – выражение, которое содержит символьные скалярные переменные. С помощью old задаются значения скалярных переменных, которые затем подставляются в символьное выражение.

Команда **snew = subs (s, new)** возвращает копию *s*, заменяя все случаи символьной скалярной переменной в *s* на new, и затем оценивает *s*. По умолчанию new = 1.

Команда **snew = subs (s)** возвращает копию *s*, заменяя символьные скалярные переменные в *s* их значениями, заданными каким-либо способом, затем оценивает *s*. Переменные без присвоенных значений остаются символьными переменными.

Вычисление значений символьных выражений проходит в два этапа. На первом этапе необходимо заменить все переменные, входящие в выражение, на их значения с помощью функции **subs**.

После выполнения функции **subs** выражение все еще остается символьным, поэтому требуется перевести полученное выражение в числовое, например с помощью функции **double**.

В качестве примера вычислим значение выражения

$$\frac{x^4 + 17x^3 + 91x^2 + 268x}{248}$$

при $x = 3$ несколькими синтаксисами. Первый способ записи:

```
syms x;
f = (x^4 + 17*x^3 + 91*x^2 + 268*x) / 248;
%заменяем все x на 3
f = subs(f, x, 3);
f =
 2163/248
f = double(f)
disp(f)
```

```
f =
   8.7218
```

Второй способ записи:

```
syms x;
f = (x^4 + 17*x^3 + 91*x^2 + 268*x) / 248;
%заменяем все x на 3
f = subs(f, 3);
f=double (f);
disp(f)
f =
   8.7218
```

Третий способ записи:

```
syms x
y = x^2;
x=2;
f=double (subs(y));
disp (f);
f =
    4
```

Если функция содержит несколько переменных, следует использовать функцию **subs** несколько раз. Так, в примере ниже функция содержит две переменные (*x* и *y*), поэтому функция **subs** применяется два раза.

```
syms x y;
f = (x+4)^4 + x^2 + (y-2)^3;
f = double(subs(subs(f, x, 2), y, 3))
f =
   1301
```

5.3. Символьное дифференцирование в MATLAB

Кратко рассмотрим еще один метод символьных вычислений – символьное дифференцирование, которое осуществляется функцией **diff**.

При вызове функции следует указать переменную, по которой будет проводиться дифференцирование. Например:

```
f = x^4 + 17*x^3 + 91*x^2 + 268*x + 248;
diff(f)
```

```
ans =
 4*x^3 + 51*x^2 + 182*x + 268
```

В этом примере функция зависит от одной переменной, поэтому производная считается по ней автоматически. Если нужно вычислить вторую производную, используем другой вариант синтаксиса:

```
f = x^4 + 17*x^3 + 91*x^2 + 268*x + 248;
diff(f, 2)
ans =
12*x^2 + 102*x + 182
```

Если символьное выражение содержит нескольких переменных, функция **diff** применяется последовательно ко всем переменным, входящим в выражение:

```
syms x y;
f = 2 * x ^ 5 + x * y + y^3;
diff(f, 'x')
ans =
10*x^4 + y
diff(f, 'y')
ans =
3*y^2 + x
```

5.4. Символьное интегрирование в MATLAB

Наряду с дифференцированием, в MATLAB можно выполнять символьное интегрирование. Иногда при анализе данных оно удобнее, чем численное интегрирование.

Символьное интегрирование в MATLAB осуществляется оператором **int**. Он выполняется так же, как и оператор дифференцирования. Например:

```
f = 4*x^3 + 51*x^2 + 182*x + 268;
int(f)
ans =
x^4 + 17*x^3 + 91*x^2 + 268*x
```

Также возможен расчет определенного интеграла. В качестве примера рассчитаем определенный интеграл на промежутке от 0 до 4:

```
f = 4*x^3 + 51*x^2 + 182*x + 268;
int (f, 0, 4)
ans =
3872
```

В MATLAB реализовано множество функций для работы с символьными вычислениями. Помимо тех, что были рассмотрены, выделим следующие функции:

– **ezplot (f)** – построение графика функции;

– **solve (f)** – решение символьных уравнений и систем;

– **taylor (f)** – разложение символьной функции в ряд Тейлора;

– **limit (f)** – вычисление предела.

Эти и многие другие функции в MATLAB имеют свои опции и параметры. Очевидно, что среда MATLAB дает широкие возможности разработчику при работе с символьными вычислениями.

Вопросы для самоконтроля

1. Как можно создать символьное выражение в MATLAB?

2. Кратко охарактеризуйте этапы вычисления значения символьного выражения.

3. Как выполняется символьное дифференцирование в MATLAB? Приведите пример.

4. Как выполняется символьное интегрирование в MATLAB? Приведите пример.

Практическая работа

ВЫЧИСЛЕНИЕ И ВЫВОД В ВИДЕ МАССИВА ТИПА ТАБЛИЦЫ ЗНАЧЕНИЯ ФУНКЦИИ, ЗАДАННОЙ АНАЛИТИЧЕСКИ

Задание. Вычислите, сохраните в виде table-массива и выведите на экран значения функции на интервале от $x_{нач}$ до $x_{кон}$ с шагом dx. Значения a, b, c, $x_{нач}$, $x_{кон}$, dx введите с клавиатуры. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий выполненным расчетам;

○ результат вычислений.

Варианты заданий

Вариант 1.

$$F = \begin{cases} ax^2 + b \text{ при } x < 0, b \neq 0; \\ \dfrac{x-a}{x-c} \text{ при } x > 0, b = 0; \\ \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

Вариант 2.

$$F = \begin{cases} \dfrac{1}{ax} - b \text{ при } x + 5 < 0, c = 0; \\ \dfrac{x-a}{x} \text{ при } x + 5 > 0, c \neq 0; \\ \dfrac{10x}{c-4} \text{ в остальных случаях.} \end{cases}$$

Вариант 3.

$$F = \begin{cases} ax^2 + bx + c \text{ при } a < 0, c \neq 0; \\ \dfrac{-a}{x-c} \text{ при } a > 0, c = 0; \\ a(x+c) \text{ в остальных случаях.} \end{cases}$$

Вариант 4.

$$F = \begin{cases} ax - c \text{ при } c < 0, x \neq 0; \\ \dfrac{x-a}{-c} \text{ при } c > 0, x = 0; \\ \dfrac{bx}{c-a} \text{ в остальных случаях.} \end{cases}$$

Вариант 5.

$$F = \begin{cases} a - \dfrac{x}{10+b} \text{ при } x < 0, b \neq 0; \\ \dfrac{x-a}{x-c} \text{ при } x > 0, b = 0; \\ 3x + \dfrac{2}{c} \text{ в остальных случаях.} \end{cases}$$

Вариант 6.

$$F = \begin{cases} ax^2 + b^2x \text{ при } c < 0, b \neq 0; \\ \dfrac{x+a}{x+c} \text{ при } c > 0, b = 0; \\ \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

Вариант 7.

$$F = \begin{cases} -ax^2 - b \text{ при } x < 5, c \neq 0; \\ \dfrac{x-a}{x} \text{ при } x > 5, c = 0; \\ \dfrac{-x}{c} \text{ в остальных случаях.} \end{cases}$$

Вариант 8.

$$F = \begin{cases} -ax^2 \text{ при } c < 0, a \neq 0; \\ \dfrac{a-x}{cx} \text{ при } c > 0, a = 0; \\ \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

Вариант 9.

$$F = \begin{cases} ax^2 + b^2x \text{ при } a < 0, x \neq 0; \\ x - \dfrac{a}{c-x} \text{ при } a > 0, x = 0; \\ 1 + \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

Вариант 10.

$$F = \begin{cases} ax^2 + b^2x + c \text{ при } x < 3, b \neq 0; \\ \dfrac{x-a}{x-c} \text{ при } x > 3, b = 0; \\ \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

Вариант 11.

$$F = \begin{cases} ax^2 + \dfrac{b}{c} \text{ при } x < 1, c \neq 0; \\ \dfrac{x-a}{x-c^2} \text{ при } x > 1{,}5, c = 0; \\ \dfrac{x^3}{c^3} \text{ в остальных случаях.} \end{cases}$$

Вариант 12.

$$F = \begin{cases} ax^2 + b^2 + c \text{ при } x < 0{,}6, b + c \neq 0; \\ \dfrac{x-a}{x-c} \text{ при } x > 0{,}6, c = 0; \\ \dfrac{x}{c} + \dfrac{x}{a} \text{ в остальных случаях.} \end{cases}$$

Вариант 13.

$$F = \begin{cases} ax^2 + b \text{ при } x - 1 < 0, b - x \neq 0; \\ \dfrac{x - a}{x} \text{ при } x - 1 > 0, b + x = 0; \\ \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

Вариант 14.

$$F = \begin{cases} -ax^2 - b \text{ при } x + c < 0, a \neq 0; \\ \dfrac{x - a}{x - c} \text{ при } x + c > 0, a = 0; \\ \dfrac{x}{c} + \dfrac{c}{x} \text{ в остальных случаях.} \end{cases}$$

Вариант 15.

$$F = \begin{cases} -ax^2 + b \text{ при } x < 0, b \neq 0; \\ \dfrac{x}{x - c} + 5{,}5 \text{ при } x > 0, b = 0; \\ \dfrac{x}{-c} \text{ в остальных случаях.} \end{cases}$$

Вариант 16.

$$F = \begin{cases} a(x + c)^2 - b \text{ при } x = 0, b \neq 0; \\ \dfrac{x - a}{-c} + 5{,}5 \text{ при } x = 0, b = 0; \\ a + \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

Практическая работа

ПРИБЛИЖЕНИЕ ФУНКЦИЙ

Приближение (аппроксимация) функции в анализе данных требуется, когда необходимо вычислить значения функции в точках разрыва, возникших по каким-либо причинам (не велась статистика, входные данные изначально неполные и т. д.), или заменить имеющуюся сложную функцию более простой.

Существует множество методов, позволяющих аппроксимировать функцию. Рассмотрим только один из них – построение интерполяционного полинома Лагранжа.

Для системы узлов x_0, x_1, …, x_n введем коэффициенты Лагранжа вида

$$L_n^i(x) = \frac{(x - x_0)(x - x_1) \dots (x - x_n)}{(x_i - x_0)(x_i - x_1) \dots (x_i - x_n)}.$$

Здесь индекс i может принимать значения 0, 1, …, n. В числителе каждый сомножитель представляет собой разность переменной x и значения одного из узлов (за исключением i-го узла, что отмечено верхним индексом в обозначении функции L_n^i).

Пример. Функция $y = f(x)$ задана таблично своими значениями в четырех узлах.

i	0	1	2	3
x_i	–1	0	2	5
$y_i = f(x_i)$	1	–3	2	4

Построим для $y = f(x)$ интерполяционный полином Лагранжа и, пользуясь им, приближенно найдем значение y в точке $x = 1$, которой нет среди узлов.

Применяя формулу, получим

$$L_3(x) = y_0 \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)} +$$

$$+ y_1 \frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)} + y_2 \frac{(x - x_0)(x - x_1)(x - x_3)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)} +$$

$$+ y_3 \frac{(x - x_0)(x - x_1)(x - x_2)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)} =$$

$$= 1 \frac{(x - 0)(x - 2)(x - 5)}{(-1 - 0)(-1 - 2)(-1 - 5)} - 3 \frac{(x + 1)(x - 2)(x - 5)}{(0 + 1)(0 - 2)(0 - 5)} +$$

$$+2 \frac{(x + 1)(x - 0)(x - 5)}{(2 + 1)(2 - 0)(2 - 5)} + 4 \frac{(x + 1)(x - 0)(x - 2)}{(5 + 1)(5 - 0)(5 - 2)} =$$

$$= -\frac{19}{45}x^3 + \frac{233}{90}x^2 - \frac{89}{90}x - 3.$$

Тогда $f(1) \approx L_3(1) = -\frac{19}{45} + \frac{233}{90} - \frac{89}{90} - 3 = -1{,}82$.

Для проверки правильности построенного полинома в MATLAB имеется ряд функций.

Функция **polyfit (x, y, n)** находит коэффициенты полинома $p(x)$ степени n.

Функция **polyval (p, x)** возвращает значение полинома степени n для заданного x.

Задание. Постройте интерполяционный полином Лагранжа для функции $f(x)$, заданной таблицей или формулой. По построенному полиному вычислите значение в заданной точке. Проверьте правильность вычислений средствами MATLAB.

Варианты заданий

Вариант 1.

x	0	1	2	3
y	–2	–5	0	–4

Вычислите значение в точке $x = 1{,}5$.

Вариант 2.

x	0	2	3
y	1	3	2

Вычислите значение в точке $x = 0{,}5$.

Вариант 3.

x	0	2	3	5
y	1	3	2	5

Вычислите значение в точке $x = 1{,}5$.

Вариант 4.

x	–1	0	1
y	4	0	1

Вычислите значение в точке $x = –0{,}5$.

Вариант 5.

x	–1	2	5	6	8	10
y	1	–2	3	7	2	–2

Вычислите значение в точке $x = 1$.

Вариант 6. Для функции $y = \ln(x)$ с узлами $x = 2; 3; 4$ вычислите значение в точке $x = 3{,}2$.

Вариант 7. Для функции $f(x) = \ln(x)$ с узлами $x = 9; 10; 12; 15$, используя значения в этих узлах, вычислите $\ln(11)$.

Вариант 8. Для функции $y = \cos \pi x$ с узлами $x = 0; 1/4; 1/3; 1/2$, используя значения в этих узлах, вычислите значение y в точке $x = 2{,}5$.

Вариант 9. Задан ряд показателей урожайности зерновых культур в целом по России (ц/га) за 1984 – 1997, 1999 гг.: 15,6; 17,6; 16,4; 15,6; 17,6; 20,3; 15,8; 18,8; 17,9; 15,6; 12,5; 14,0; 17,8; 10,4; 10,6. Вычислите показатели урожайности в 1998 г.

Вариант 10.

x	–3	–1	2
y	–5	–11	10

Вычислите значение в точке $x = 0$.

Вариант 11. По значениям полинома третьей степени $P_3(-1) = -11$, $P_3(1) = -3$, $P_3(2) = 1$, $P_3(3) = 13$ постройте полином Лагранжа и вычислите значение в точке $x = 1{,}8$.

Вариант 12. Данные о численности населения и количестве трудоспособных за период с 2006 по 2011 г. представлены с сайта https://rosinfostat.ru/ekonomicheski-aktivnoe-naselenie/. Вычислите значение показателя численности занятых для 2012 г.

Год	Всего, тыс. чел.	Занятые, тыс. чел.
2006	74 419	69 169
2007	75 289	70 770
2008	75 700	71 003
2009	75 694	69 410
2010	75 478	69 934
2011	75 779	70 857

Вариант 13. Данные о численности населения и количестве трудоспособных за период с 2006 по 2011 г. представлены с сайта https://rosinfostat.ru/ekonomicheski-aktivnoe-naselenie/ (см. данные варианта 12). Вычислите значение показателя численности населения для 2012 г.

Вариант 14. Данные о численности населения и количестве трудоспособных за период с 2006 по 2011 г. представлены с сайта https://rosinfostat.ru/ekonomicheski-aktivnoe-naselenie/ (см. данные варианта 12). Вычислите значение показателя численности населения для 2013 г.

Вариант 15. Данные о численности населения и количестве трудоспособных за период с 2012 по 2017 г. представлены с сайта https://rosinfostat.ru/ekonomicheski-aktivnoe-naselenie. Вычислите значение показателя численности населения для 2018 г.

Год	Всего, тыс. чел.	Занятые, тыс. чел.
2012	75 677	71 545
2013	75 529	71 391
2014	75 428	71 539
2015	76 588	72 324
2016	76 636	72 393
2017	76 285	72 316

Вариант 16. Дана таблица значений некоторой функциональной зависимости, полученной из $n = 10$ опытов. Вычислите значение в точке $x = 6{,}5$.

x	1	2	3	4	5	6	7	8	9	10
y	1,0	1,5	3,0	4,5	7,0	8,5	8,0	11,3	1,0	9,5

Глава 6
ГРАФИЧЕСКАЯ ВИЗУАЛИЗАЦИЯ ВЫЧИСЛЕНИЙ В MATLAB. ВОЗМОЖНОСТИ ДВУМЕРНОЙ ГРАФИКИ

В MATLAB имеется три функции построения двумерных графиков: **plot ()**, **fplot ()** и **ezplot ()**.

Для создания подписей графиков, осей и отображения сетки на графике используются следующие функции языка MATLAB, перечисленные ниже.

grid [on, off]	Включает/выключает сетку на графике
title (‘заголовок графика’)	Создает надпись заголовка графика
xlabel (‘подпись оси Ox’)	Создает подпись оси *Ox*
ylabel (‘подпись оси Oy’)	Создает подпись оси *Oy*
text (x, y, ’текст’)	Создает текстовую надпись в координатах (*x*, *y*)

6.1. Функция plot

Это функция для построения 2D-графиков. Синтаксис следующий:

plot (X, Y)
plot (X, Y, LineSpec)
plot (X1, Y1, ..., Xn,Yn)
plot (X1, Y1, LineSpec1, ..., Xn, Yn, LineSpecn)
plot (Y)
plot (Y, LineSpec)
plot (___, Name, Value)
plot (ax, ___)
h = plot (___)

Функция **plot (X, Y)** создает 2D-график функции, где *X* – аргумент функции, задаваемой в виде скаляра или вектора; *Y* – функция, представленная в аналитическом виде или в виде вектора или матрицы.

Если обе переменные *X* и *Y* – векторы или матрицы, у них должны быть равные длина или размер. В противном случае данные отображаются на разных графиках.

Если одна из переменных (*X* или *Y)* – вектор, а другая – матрица, одна из размерностей матрицы должна равняться длине вектора.

Если длина вектора равна количеству строк матрицы, то для значений вектора функцией **plot** в виде графиков будет отдельно выведен каждый столбец матрицы.

В примере ниже для значений 2, 3, 4, 5 вектора *X* выводятся четыре графика, состоящие из значений столбцов матрицы *Y*. Соответственно 1 6 10; 2 7 11; 3 8 12; 5 9 13 (рис. 6.1):

```
x=[2 3 4 5];
y=[1 2 3 5;6 7 8 9; 10 11 12 13];
plot(x,y)
```

Аналогично если длина вектора равна количеству столбцов матрицы, то для значений вектора функцией **plot** в виде графиков будет отдельно выведена каждая строка матрицы.

Например, вектор *X* имеет две строки и три столбца. Соответственно, функцией **plot** для значений вектора *X* будет выведено две строки матрицы (рис. 6.2):

```
x=[2 3 4];
y=[1 2; 6 7; 10 11];
plot(x,y)
```

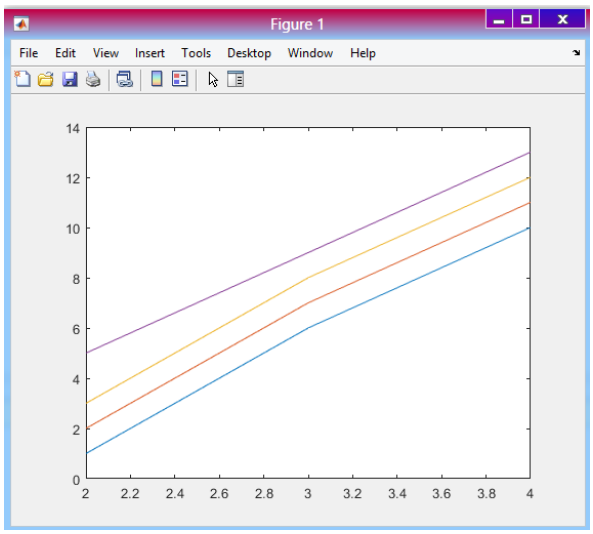


Рис. 6.1. Вид графика, если длина вектора равна количеству строк матрицы

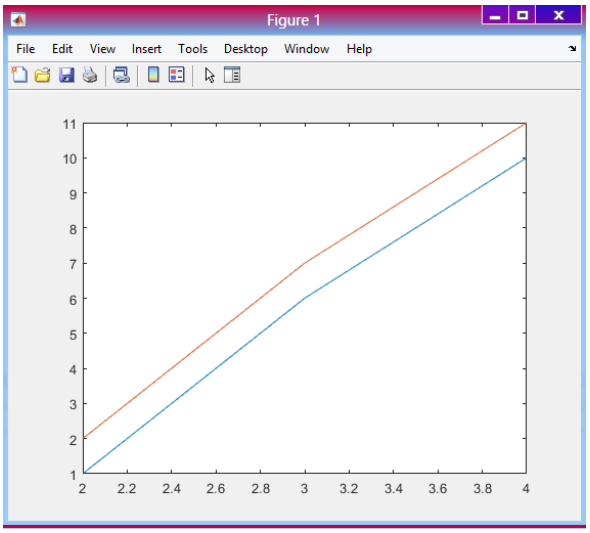


Рис. 6.2. Вид графика в случае равенства длины вектора количеству столбцов матрицы

Если матрица квадратная, то в виде графиков функций будет представлен каждый столбец матрицы (рис. 6.3). Например:

```
x=[2 3 4];
y=[1 2 4; 6 7 8; 10 11 12];
plot(x,y)
```

Если, например, *X* – скаляр, а *Y* – скаляр или вектор, функция **plot** выведет дискретные точки графиков функций. В этом случае, чтобы видеть точки, необходимо задать символ маркера, например plot (X, Y, 'O') (рис. 6.4).

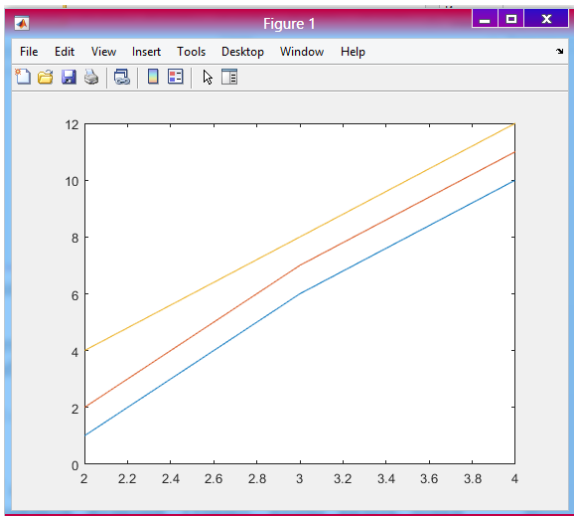


Рис. 6.3. Вид графика в случае квадратной матрицы

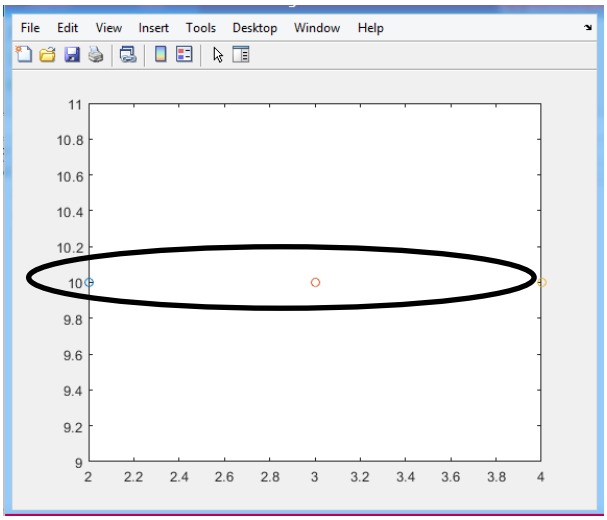


Рис. 6.4. Вид графика в случае, когда один параметр – скаляр, а другой – вектор

Для функции $y = f(x)$, заданной в аналитическом виде, ниже приведен рисунок и пример команд построения графика функции $y = x + 3$ для x в диапазоне [0; 1] с постоянным шагом $h = 0{,}2$ (рис. 6.5):

```
x = 0: 0.2 :1;
y = x + 3;
plot (x,y)
```

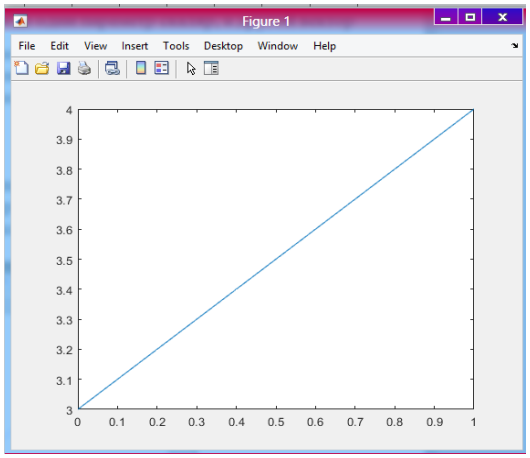


Рис. 6.5. Вид графика в случае аналитического задания функции

Функция **plot (X, Y, LineSpec)** аналогична функции **plot (x, y)** и отличается лишь наличием вектора констант LineSpec, определяющего цвет линий графика, тип точек и линий функции. Стили графиков приведены в таблице.

Стили графиков

Тип точки		Цвет линии		Тип линии	
.	Точка	Y	Желтый	-	Сплошная
O	Окружность	M	Фиолетовый	:	Двойной пунктир
x	Крест	C	Голубой	-.	Штрихпунктир
+	Плюс	R	Красный	--	Штриховая
*	Звездочка	G	Зеленый		
S	Квадрат	B	Синий		
D	Ромб	W	Белый		
^,v,>,<	Треугольник вверх (вниз, влево, вправо)	K	Черный		
P	Пятиугольник				
H	Шестиугольник				

При задании стиля символ Linespec представляется в виде вектора или строки, элементы которых – тип точки, цвет линии и тип линии. В первом случае параметры разделены запятыми и выделены одиночными кавычками. Например:

plot (x, y, ['R', '*', '-.']);

Это график красного цвета (R), точки графика представлены в виде звездочек (*), линии штрихпунктирные (-.). Во втором случае значения параметров также выделены одиночными кавычками и разделены пробелами. Например:

plot (x, y, 'R + -.').

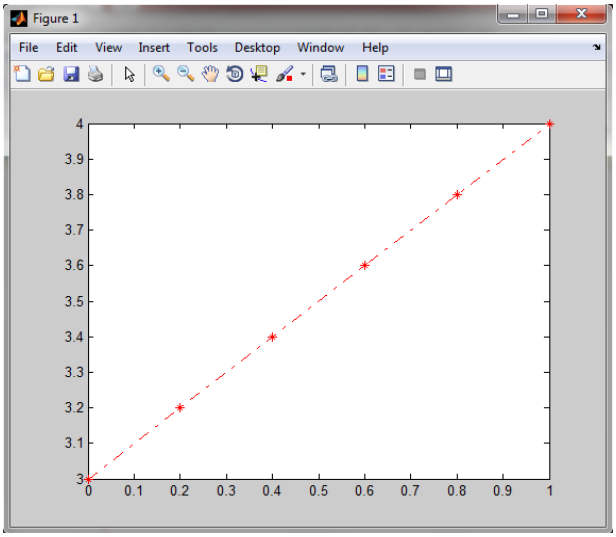


Рис. 6.6. График функции, выполненный с использованием стилей

На рисунке 6.6 показан график функции $y = 3 + x$, выполненный в этом стиле. Соответствующие команды представлены ниже.

Первый способ

```
x = 0: 0.2 :1;
y = x + 3;
plot (x, y, ['R', '*', '-.'])
```

Второй способ

```
x = 0: 0.2 :1;
y = x + 3;
plot (x, y, 'R  +  -.')
```

Функция **plot (X1, Y1, ..., Xn, Yn)** позволяет строить большое число математических функций на одном графике. Обозначения имеют следующий смысл:

x_i – i-й массив аргументов, заданный в виде вектора;

y_i – i-й массив значений функции для заданного массива аргументов (может задаваться и в аналитическом виде).

В качестве примера построим графики функций sin и cos для 100 значений $x \in [-2\pi; 2\pi]$ в одном и том же окне figure (рис. 6.7):

```
% 100 линейно распределенных значений между –2π; 2π
x = linspace(-2*pi,2*pi);
y1 = sin(x);
y2 = cos(x);
%вызов окна визуализации графиков figure
figure
plot(x,y1,x,y2)
```

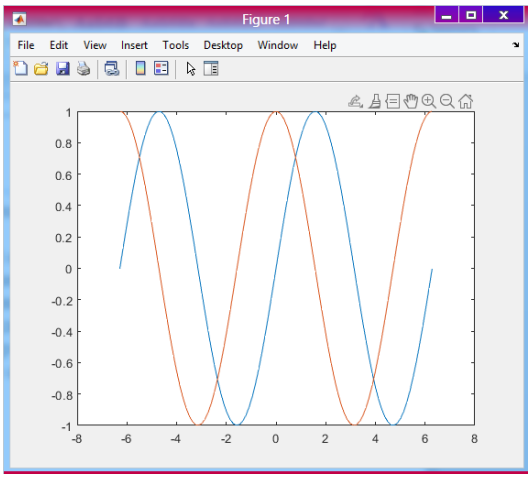


Рис. 6.7. Графики двух функций в одной координатной плоскости

Функция **plot (X1, Y1, LineSpec1, ..., Xn, Yn, LineSpecn)** устанавливает стиль линии, тип маркера и цвет для каждой линии.

LineSpec$_i$ – стиль графика для i-й функции. Если стиль не задан, система MATLAB задает его автоматически.

В качестве примера построим три синусоиды с небольшим сдвигом фазы, равным 0,25, между каждой линией. Для первой линии используем стиль линии, заданный по умолчанию. Для второй линии зададим стиль «пунктирная линия» и для третьей линии – стиль «точечная линия» (рис. 6.8). Например:

```
x = 0:pi/100:2*pi;
y1 = sin(x);
y2 = sin(x-0.25);
y3 = sin(x-0.5);
figure
plot(x,y1,x,y2,'--',x,y3,':')
```

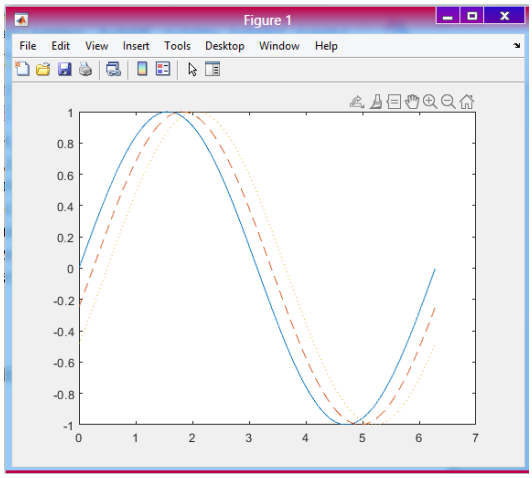


Рис. 6.8. Графики синусоид с разным стилем оформления

При необходимости можно задать заголовок, надписи осей и коор-

динатную сетку, записав последовательно команды: **title** (‘текст заголовка’), **xlabel** (‘надпись оси OX’), **ylabel** (‘надпись оси OY’), **grid on**. Соответствующие надписи будут последовательно появляться на графике функции. Полный текст программы приведен ниже:

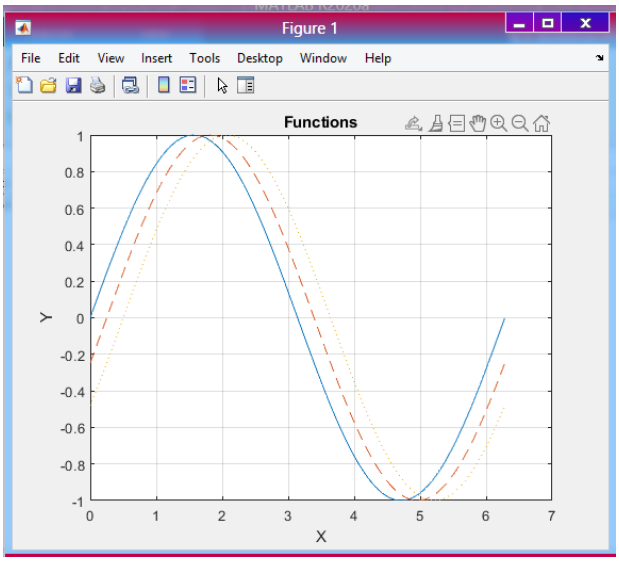


Рис. 6.9. Графики функций
с заголовком, подписями осей
и координатной сеткой

```
x = 0:pi/100:2*pi;
y1 = sin(x);
y2 = sin(x-0.25);
y3 = sin(x-0.5);
figure
plot(x,y1,x,y2,'--',x,y3,':')
title ('Functions');
xlabel ('X');
ylabel ('Y');
grid on;
```

На рисунке 6.9 показан график функции, при построении которого использованы указанные выше команды.

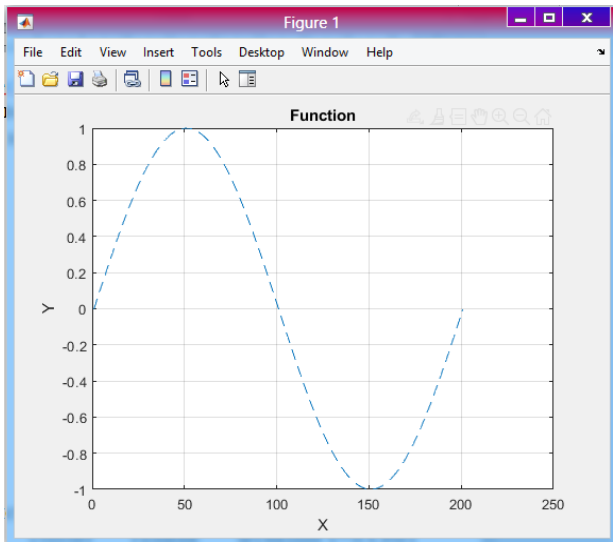


Рис. 6.10. График функции,
построенный с помощью функции
plot (Y, LineSpec)

Функция **plot (Y, LineSpec)** устанавливает стиль линии, символ маркера и цвет. Аналогична функции **plot (X, Y, LineSpec)**. Пример кода и визуализация графика функции представлены ниже (рис. 6.10):

```
x = 0:pi/100:2*pi;
y = sin(x);
figure
plot(x,y)
plot (y,'--');
title ('Function');
xlabel ('X');
ylabel ('Y');
grid on;
```

Функция **plot (___, Name, Value)** задает свойства линии с помощью одной или нескольких пар Name, Value (имя – значение). Эта опция используется с любыми комбинациями входных аргументов

предыдущих синтаксисов. Настройки пары «имя – значение» применяются ко всем построенным графикам.

Список и описание некоторых свойств представлены ниже (полный список свойств представлен в документации [15]):

– LineWidth 'LineWidth' задает ширину линии в виде положительного значения в точках, где одна точка составляет 1/72 дюйма. Значение по умолчанию составляет 0,5. Ширина линии не может быть меньше ширины пикселя;

– MarkerSize 'MarkerSize' задает размер маркера в виде положительного значения в точках, где одна точка составляет 1/72 дюйма. Значение по умолчанию составляет 6;

– MarkerFaceColor и MarkerEdgeColor задают цвет внутри маркера и цвет контура маркера в виде триплета RGB, представляющего собой вектор-строку, элементы которого определяют интенсивность красных, зеленых и синих компонентов цвета. Интенсивность должна быть в области значений [0; 1]. Например, [0,4 0,6 0,7].

В качестве примера построим график функции

$y = \mathrm{tg}(\sin(x)) - \sin(\mathrm{tg}(x))$.

Используя опцию LineSpec, зададим пунктирную зеленую линию с квадратными маркерами. С помощью пары Name,Value зададим ширину линии (LineWidth), размер (MarkerSize) и цвет обводки маркера (MarkerEdgeColor). Поверхность маркера (MarkerFace Color) зададим значениями цветов модели RGB (рис. 6.11):

```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
figure
plot(x,y,'--gs',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','b',...
    'MarkerFaceColor',[0.5,0.5,0.5])
```

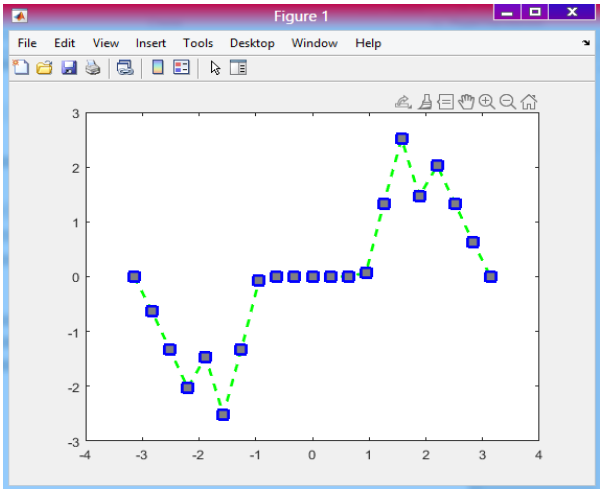


Рис. 6.11. Внешний вид графика функции, заданный с помощью опции LineSpec

Функция **plot (ax, ___)**: с помощью опции ax в текущей системе координат задаются другие оси графика. Опция ax может

предшествовать любой из комбинаций входных аргументов из предыдущих синтаксисов.

Начиная с версии R2019b, можно отобразить плиточное размещение графиков с помощью функций **tiledlayout** и **nexttile**.

В примере ниже с помощью функции **tiledlayout** создается мозаичное размещение графиков с размерами 2 × 1.

С помощью функции **nexttile** создаются два ax-объекта – оси координат. Заголовки и метки оси *Y* графиков задаются функциями **title** и **ylabel** (рис. 6.12):

```
% задание и размещение двух графиков в один столбец
x = linspace(0,3);
y1 = sin(5*x);
y2 = sin(15*x);
tiledlayout(2,1)
% Top plot
ax1 = nexttile;
plot(ax1,x,y1)
title(ax1,'Top Plot')
ylabel(ax1,'sin(5x)')
% Bottom plot
ax2 = nexttile;
plot(ax2,x,y2)
title(ax2,'Bottom Plot')
ylabel(ax2,'sin(15x)')
```

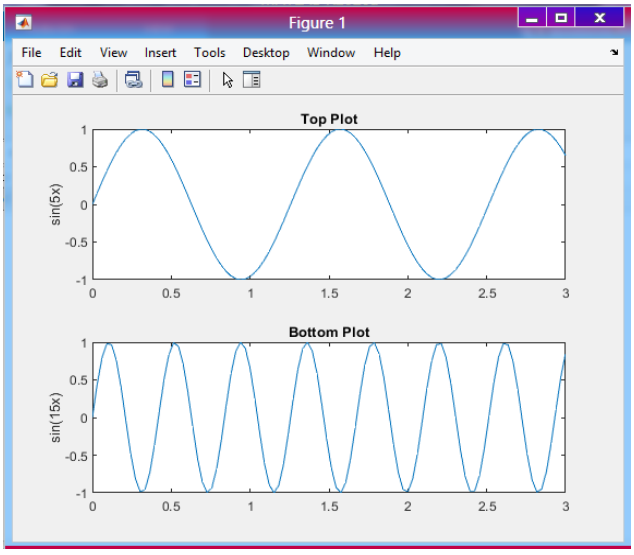
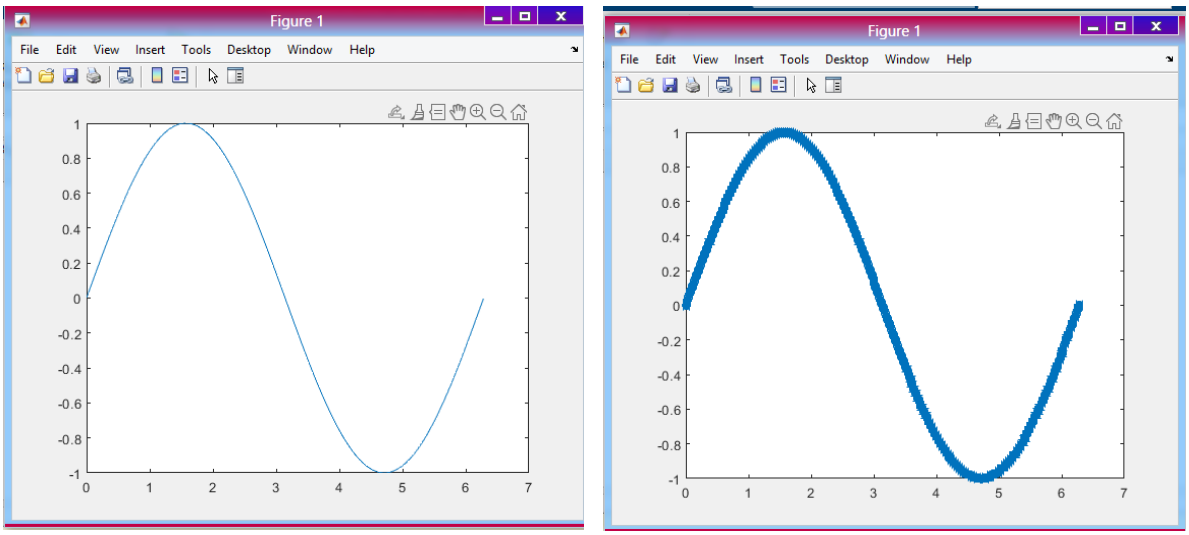


Рис. 6.12. Визуализация графиков с помощью функции **plot (ax,___)**

Функция **h = plot (___)** отличается от предыдущих синтаксисов тем, что с помощью объекта *h* можно изменять свойства линий на графике после создания графика.

В качестве примера визуализируем график функции sin (*x*), а затем изменим внешний вид графика с помощью объекта *h* (рис. 6.13):

```
x = 0:pi/100:2*pi;
y = sin(x);
h=plot(x,y)
%Изменение свойств графика
x = 0:pi/100:2*pi;
y = sin(x);
h=plot(x,y)
h.LineWidth = 2;
h.Marker = '*';
```



а) *б)*

Рис. 6.13. Внешний вид графика: *а* – до изменения его свойств; *б* – измененный вид графика

6.2. Функция fplot ( )

Функция **fplot ( )** – модификация функции **plot**. Она строит график функции $y = f(x)$ без предварительного вычисления векторов ($x1$, $x2$, …) и ($y1$, $y2$, ...), т. е. по их символьному обозначению.

ВАЖНО! Предварительно необходимо убедиться в наличии программного обеспечения Symbolic Math Toolbox.

По сравнению с функцией **plot** функция **fplot** берет на себя вычисление таблицы значений функции, проявляя при этом некоторый интеллект: в местах резкого изменения функции значения аргумента *x* выбираются с более мелким шагом.

Синтаксис следующий:

```
fplot (f)
fplot (f, [xmin xmax]) или fplot (f, xinterval)
fplot (xt, yt) или fplot (funx, funy)
fplot (xt, yt, [tmin tmax]) или fplot (funx, funy, tinterval)
fplot (___, LineSpec)
fplot (___, Name,Value)
fplot (ax,___)
fp = fplot (___)
```

Функция **fplot (f)** строит график функции *f* по ее символьному обозначению на интервале по умолчанию [–5; 5].

В качестве примера построим график функции tg(*x*) на интервале по умолчанию [–5; 5]. Соответствующие код и график представлены ниже (рис. 6.14):

```
syms x;
fplot (tan (x))
```

В примере ниже показан другой вариант символьного обозначения функции (рис. 6.15):

```
syms f(x);
f (x) = cos (x);
fplot (f)
```

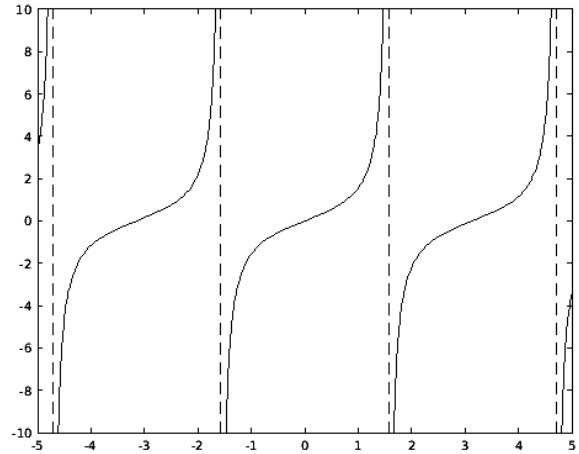


Рис. 6.14. График функции tg(*x*), построенный по ее символьному обозначению

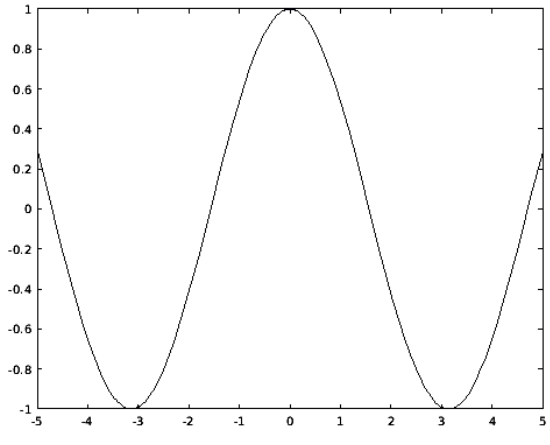


Рис. 6.15. График функции cos(*x*)

Функции **fplot (f, [xmin xmax])** или **fplot (f, xinterval)** используется тогда, когда график функции необходимо построить на интервале, отличном от интервала по умолчанию. Начало и конец нового интервала задаются двухкомпонентным вектором [xmin xmax].

В качестве примера построим график функции $y = e^x$ на интервале $[-\pi/2; \pi/2]$. Программный код и график представлены ниже (рис. 6.16):

```
fplot(@(x) exp(x),[-3 0],'b')
```

Функции **fplot (xt, yt)** или **fplot (funx, funy)** используются для построения параметрических кривых $xt = x(t)$ и $yt = y(t)$ на интервале t [–5; 5] (по умолчанию) (рис. 6.17):

```
xt = @(t) cos(3*t);
yt = @(t) sin(2*t);
fplot(xt,yt)
```

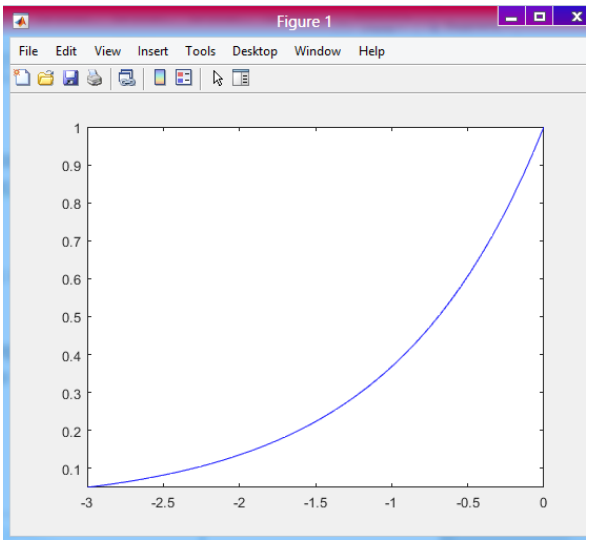


Рис. 6.16. График функции $y = e^x$

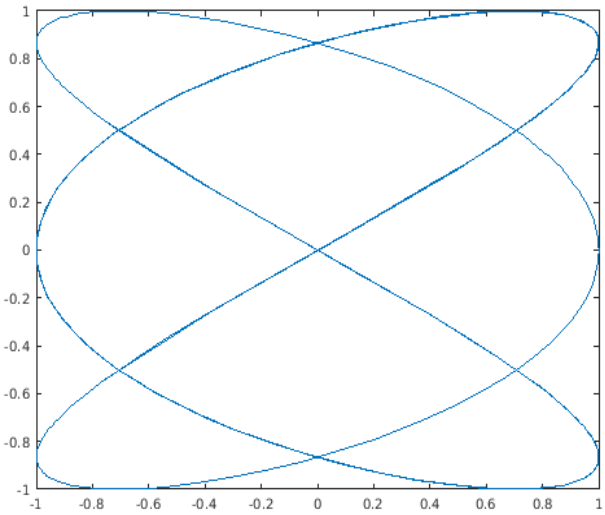


Рис. 6.17. График параметрических кривых на интервале по умолчанию

Функции **fplot (xt, yt, [tmin tmax])** или **fplot (funx, funy, tinterval)** отличаются от предыдущих тем, что используются для построения параметрических кривых $xt = x(t)$ и $yt = y(t)$ (рис. 6.18) на заданном интервале [tmin; tmax], отличном от интервала по умолчанию. Пример кода и график функции представлены ниже:

```
xt = @(t) cos(3*t);
yt = @(t) sin(2*t);
fplot(xt,yt,[-1 1])
```

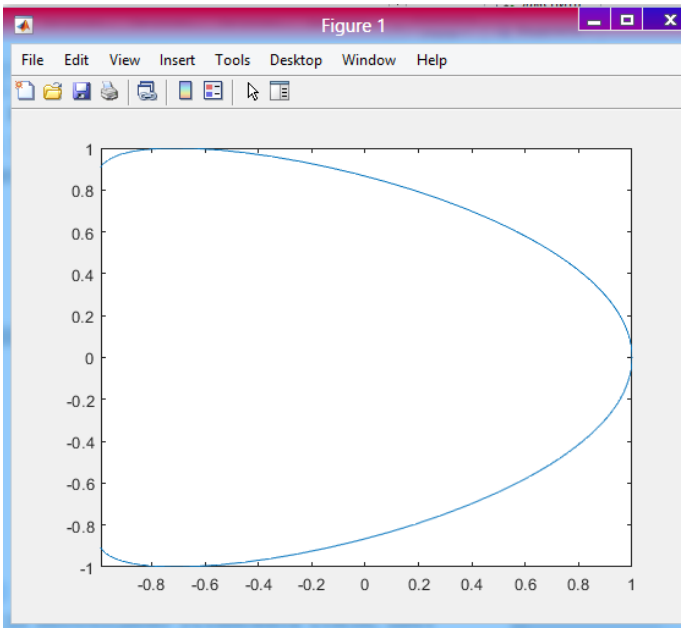


Рис. 6.18. График параметрических кривых
на интервале [–1; 1]

Функция **fplot (___, LineSpec)** применяется в том случае, когда необходимо установить стили, цвет линии и маркера графика. Используется опция LineSpec, описанная выше.

В качестве примера построим три синусоиды со сдвигом фазы между каждой линией.

Для первой линии используем ширину линии, равную двум (linewidth = 2). Для второй зададим пунктирный стиль красной линии с круговыми маркерами. Для третьей – голубой стиль штрих-пунктирной линии с маркерами-звездочками. Отобразим легенду (рис. 6.19):

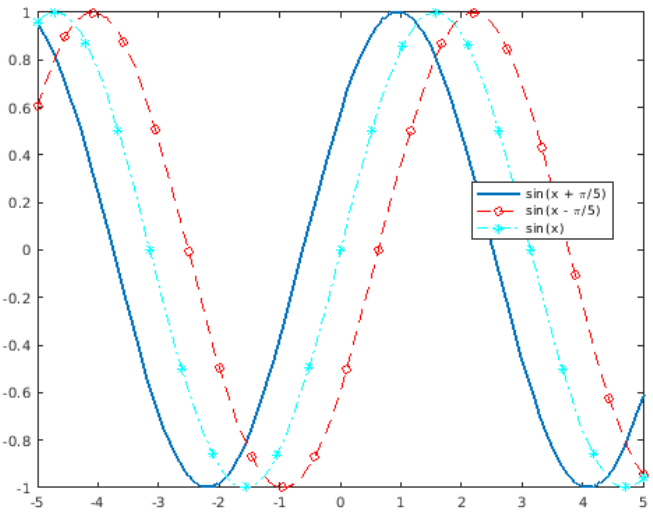


Рис. 6.19. Графики функций,
построенные с помощью функции
fplot (___, LineSpec)

```
syms x
fplot (sin(x + pi/5),  'Linewidth', 2)
hold on
fplot (sin (x - pi/5), '--or')
fplot (sin (x), '-.*c')
legend ('show', 'Location', 'best')
hold off
```

Функция **fplot (___, Name, Value)** задает свойства линии

с помощью одной или нескольких пар аргументов Name, Value. Эту опцию можно использовать с любыми комбинациями входных аргументов из предыдущих синтаксисов.

Парные настройки Name, Value применяются ко всем построенным графикам. Чтобы установить опции для отдельных линий, необходимо использовать объекты fp (fp = fplot (___)).

В качестве примера зададим толщину линии графика функции sin(x + π/2) равной трем. График функции и код представлены ниже (рис. 6.20):

```
syms x
fplot (sin(x + pi/5),  'Linewidth', 3)
hold on
```

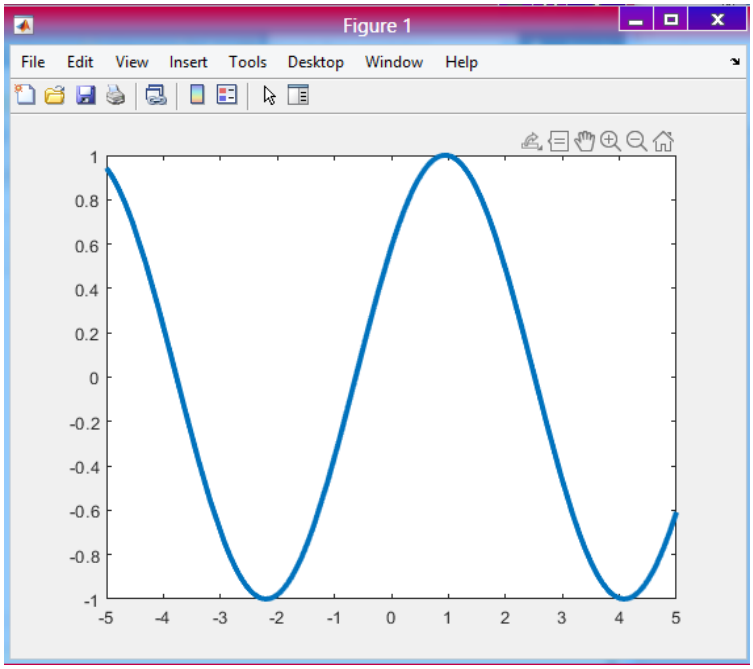


Рис. 6.20. График функции sin(x + π/2)
с толщиной линии, равной трем

Функция **fplot (ax, ___)** или ее аргументы заданы через указатель ax вместо текущей системы координат *XOY* (рис. 6.21, 6.22). Например:

```
fplot(@(x) sin (x))
```

@(x) – указатель на значения вектора x, являющегося аргументом функции. В примере выше значения x заданы по умолчанию, следовательно, принадлежат интервалу [–5; 5] (рис. 6.21).

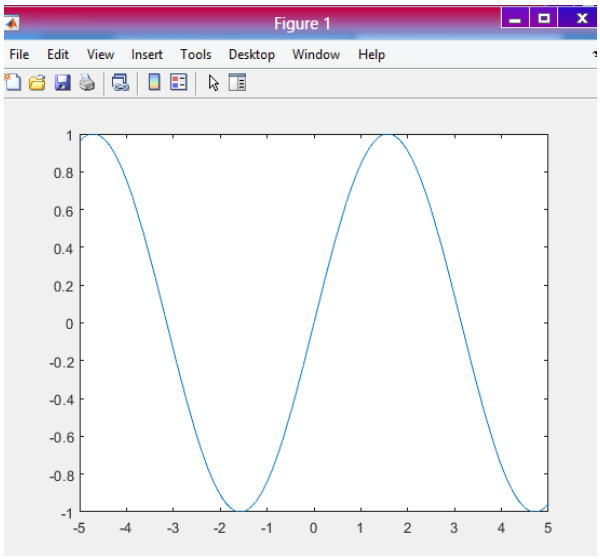


Рис. 6.21. График функции, аргумент которой задан через указатель

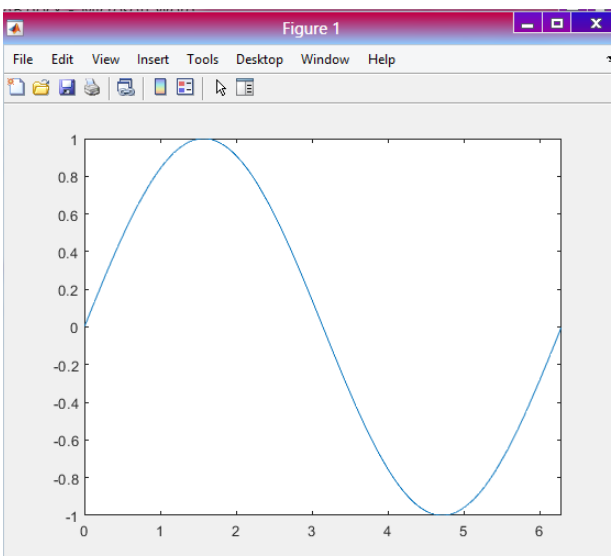


Рис. 6.22. График функции, обращение к которой задано через указатель

Для другого интервала значений *x* следующий пример:

```
syms x;
fplot(@sin, [0 2*pi])
```

Команда **fp = fplot (___)** в зависимости от типа графика возвращает функциональный объект линии или параметрированный объект линии. Используется в случае, если необходимо изменить свойства определенной линии.

В качестве примера выведем график функции sin в виде линии, толщина которой равна двум, в виде двоеточий красного цвета и синими маркерами в виде символа «x». Пример кода и соответствующий график представлены ниже (рис. 6.23):

```
fp=fplot(@(x) sin(x), 'Linewidth',2);
fp.LineStyle = ':';
fp.Color = 'r';
fp.Marker = 'x';
fp.MarkerEdgeColor = 'b';
```

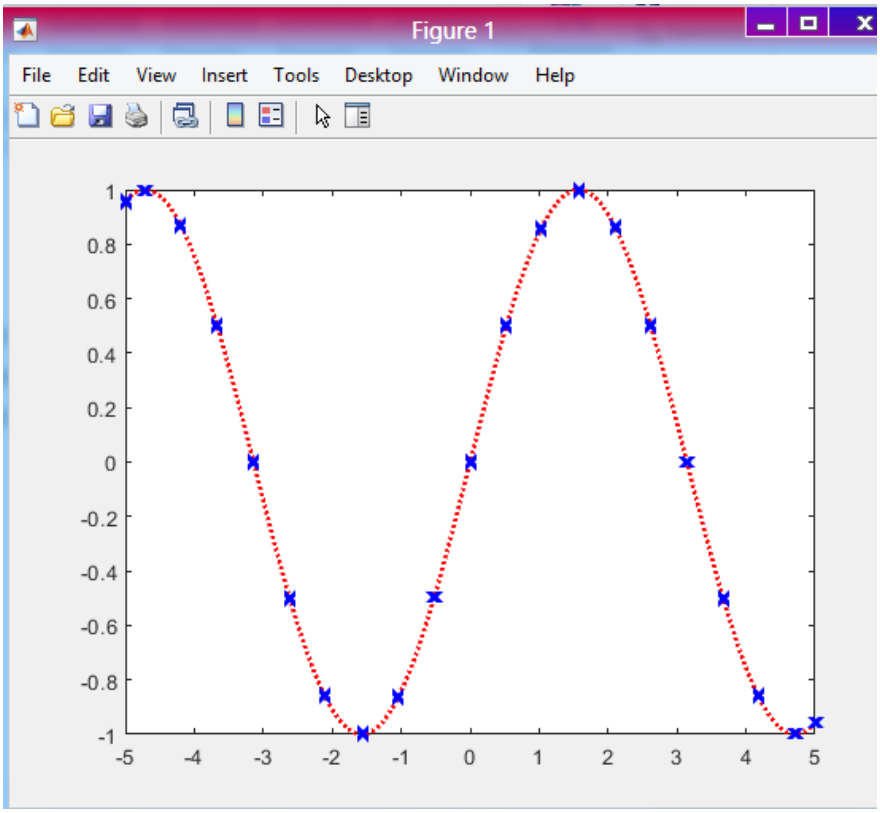


Рис. 6.23. Изменение графика функции
с помощью объекта fp

6.3. Функция fimplicit ( )

Применяется для рисования графиков функций, заданных символически в виде уравнений. В ранних версиях для этой цели использовалась функция **ezplot ()**. Синтаксис и аргументы аналогичны функциям plot и fplot.

Приведем несколько примеров.

Построим с помощью функции **fimplicit** график гиперболы $x^2 - y^2 = 1$ на интервале по умолчанию. Отличие от функций **plot** и **fplot** – использование интервала по умолчанию для обеих переменных x и y. Программный код и внешний вид графика представлены ниже (рис. 6.24):

```
syms x y
fimplicit(x^2 - y^2 == 1)
```

Следующий пример – график половины круга $x^2 + y^2 = 3$ на интервалах $-4 < x < 0$ и $-2 < y < 2$. Интервалы графического вывода задаются в качестве второго аргумента функции **fimplicit**. Код и внешний вид окна программы представлены ниже (рис. 6.25):

```
syms x y
circle = x^2 + y^2 == 3;
fimplicit(circle, [-4 0 -2 2])
```

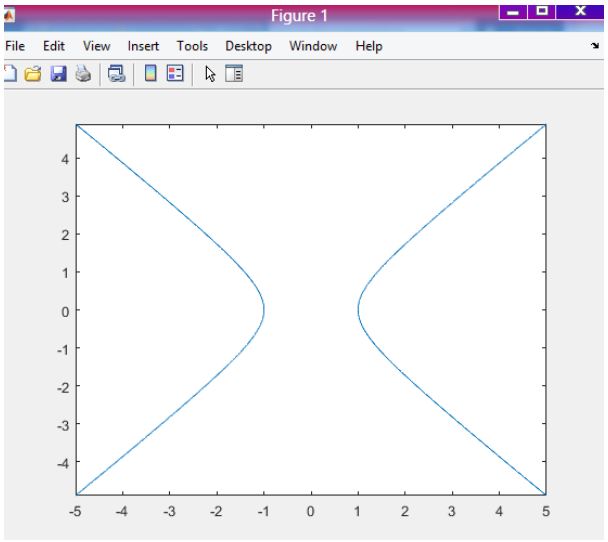


Рис. 6.24. График гиперболы, построенный с помощью функции **fimplicit**

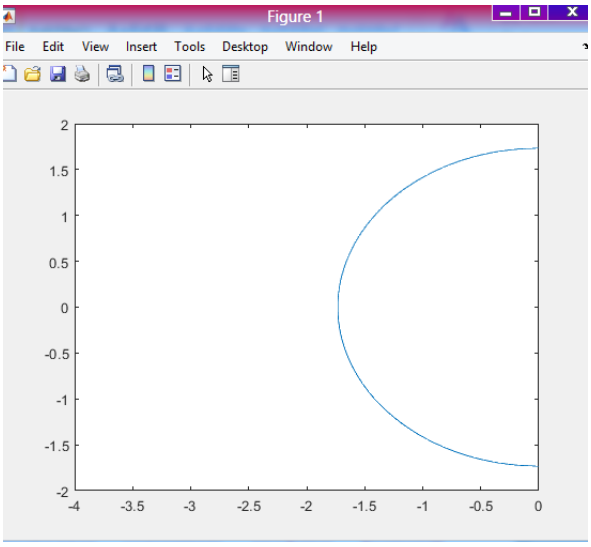


Рис. 6.25. График половины круга, построенный с помощью функции **fimplicit**

ВАЖНО! Описанные выше функции задают разные способы построения графика. MATLAB позволяет совмещать графики функций, построенные разными способами, с помощью команды **hold**, реализующей переключение от одного режима к другому:

– команда **hold on** включает режим сохранения текущего графика, так что последующие команды приведут к добавлению новых графиков в графическом окне;

– команда **hold off** выключает режим сохранения графика.

Вопросы для самоконтроля

1. Перечислите способы построения графиков функций и их различия.

2. Перечислите и дайте краткую характеристику основным стилям линии, маркера и цвета.

3. Как построить график функции, заданной параметрически?

4. С помощью какой команды можно отобразить сетку в координатных осях?

5. Как построить несколько графиков в одной координатной плоскости с помощью одной команды (с помощью нескольких команд)?

6. С помощью какой команды можно вывести текстовую надпись в графическое окно?

7. Для чего используют логарифмический масштаб в графиках? Приведите примеры.

8. Что такое легенда и когда ее используют?

Практическая работа

ПОСТРОЕНИЕ ГРАФИКОВ ФУНКЦИЙ В СРЕДЕ MATLAB

Задание. Постройте график функции с помощью функций **plot** и **fplot**. Оформите отчет.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого варианта:

○ текст задания с указанием варианта;

○ скрипт, соответствующий выполненным расчетам и построениям;

○ скриншот окна *Figure* с графиком(ами) функции(й);

– для каждого графика задать: пределы отображения, заголовок, подписи, легенду, подписи осей, тип, цвет линии и маркера;

– несколько графиков размещаются в одном окне *Figure* один под другим;

– если в аналитической записи функции присутствует логарифм, график строится двумя способами: в обычном и логарифмическом масштабах.

Варианты заданий

Вариант 1. $y = x^3 - 2x^2 - 7x + 4$.

Вариант 2. $y = x\ln^2(x)$.

Вариант 3. $y = \frac{x^3}{1+x^2}$.

Вариант 4. В результате эксперимента получена матрица значений *A* с размерами 3 × 4. Получите вектор *C*, состоящий из элементов второй строки матрицы. Отобразите элементы массива *C* в виде графика.

Вариант 5. $y_1 = \cos(x), y_2 = \sin(x), y_3 = x^{1/2}$.

Вариант 6. Постройте график зависимости времени возведения матрицы в квадрат от ее порядка. Матрицу заполните случайными целыми числами в диапазоне [1; 10].

Вариант 7. Отобразите данные, представленные в таблице, в виде графика(ов).

Количество	Эксперимент 1	Эксперимент 2	Эксперимент 3
10^3	0,2755529361166	0,04623873215974	0,06307710110520
10^4	0,28666151628344	0,67483306464728	0,85628458136542
10^5	2,84742674641796	4,92785226951551	6,29249360982561
10^6	28,5048610591205	48,4353558734875	60,7404539333518

Вариант 8. $y = \sqrt{\ln^2(x) - 1}$.

Вариант 9. $y = \ln|1 - x^2|$.

Вариант 10. $y = \cos(3x) + \mathrm{e}^{-x/2}$.

Вариант 11. Даны оценки, полученные выпускниками одной из школ за сочинение: 3/4; 4/3; 3/1; 3/3; 2/3; 3/3; 5/3; 4/4; 4/2; 5/5; 2/3; 5/4; 2/3; 3/3; 4/5. Выставлялись две оценки: первая – по литературе; вторая – по русскому языку. Представьте оценки в виде двух векторов *L* и *R*. Отобразите элементы векторов в графическом виде.

Вариант 12. $y = \sin(x) \cdot \mathrm{e}^x$.

Вариант 13. $y = -\cos\left(x + \frac{\pi}{3}\right) + 1{,}5$.

Вариант 14. $y = |x| + |x - 1| + \cdots + |x - n|,\ n \in Z$.

Вариант 15. $y = (3x)^{1/2}\lg(2x)^3$.

Вариант 16. $y = \lg(x^2 - 1)x^{-1/2}$.

Глава 7
ИМПОРТ И ЭКСПОРТ ДАННЫХ В MATLAB

7.1. Форматы файлов для импорта и экспорта, поддерживаемые в MATLAB

В таблице представлены форматы файлов, которые можно импортировать в MATLAB и экспортировать из него, а также соответствующие им функции импорта (экспорта).

Основные форматы файлов

Содержимое файла	Расширение	Описание	Функция импорта	Функция экспорта
Отформатированные данные MATLAB	MAT	Сохраненное рабочее пространство MATLAB	load	save
Текст	Любое, включая csv и txt	Запятая – разграничитель чисел. Соединение текста и чисел	readmatrix	writematrix
		Разграничение чисел или текста и чисел, ориентированное на столбец	readtable readcell readvars	writetable writecell
Электронная таблица	XLS XLSX XLSM XLSB (системы с Microsoft Excel только для Windows) XLTM (только импорт) XLTX (только импорт) ODS (системы с Microsoft Excel только для Windows)	Данные расположены на рабочем листе в столбцах или в области значений электронной таблицы	readmatrix readtable readcell readvars	writematrix writetable writecell

Окончание

Содержимое файла	Расширение	Описание	Функция импорта	Функция экспорта
Расширяемый язык разметки	XML	XML-форматированный текст	xmlread	xmlwrite
Научные данные	cdf	Специальный формат для сохранения научных данных	cdfread	cdflib
	fits	Гибкая система передачи изображения	fitsread	fitswrite
	hdf	Иерархический формат данных	imread	imwrite
	h5	Специализированный формат для хранения научных данных в файлах	h5read	h5write
	nc	Сетевая форма общих данных (NetCDF)	ncread	ncwrite
Изображение	bmp gif jpeg jpg jp2 jpf jpx j2c j2k	Форматы графических данных	imread	imwrite
	png	Сетевая графика		
Аудио	aiff aifc flac ogg wav	Форматы аудиофайлов	audioread	audiowrite
	avi	Формат файлов, в которых аудио и видео чередуются		

7.1.1. MAT-файлы. Функция load

MAT-файлы содержат данные в двоичном формате. Это могут быть переменные, функции, массивы, строки символов, аудио- и видеоформаты и т. д.

Функция **load** загружает данные из файла в рабочую область. Синтаксис следующий:

```
load (filename)
load (filename, variables)
load (filename, '-ascii')
load (filename, '-mat')
load (filename, '-mat', variables)
S = load (___)
load filename
```

Функция **load (filename)** считывает данные из файла с именем filename. Если filename – MAT-файл, функция **load (filename)** загружает данные в рабочую область MATLAB. Если filename – ASCII-файл, функция **load (filename)** создает массив двойной точности, содержащий данные из файла.

Функция **load (filename, variables)** загружает данные из MAT-файла filename в заданные переменные variables.

Функция **load (filename, '-ascii')** загружает данные из файла filename как ASCII-файл независимо от расширения файла.

Функция **load (filename, '-mat')** загружает данные из файла filename как MAT-файл независимо от расширения файла.

Функция **load (filename, '-mat', variables)** загружает данные из файла filename как MAT-файл независимо от расширения файла в заданные переменные variables.

Команда **S = load (___)** загружает данные из файла в переменную *S*, форма параметра функции может быть указана любым перечисленным выше способом. Если filename – MAT-файл, то *S* – массив структур. Если filename – ASCII-файл, то *S* – массив двойной точности, содержащий данные из файла.

Рассмотрим примеры применения этой функции.

Загрузим содержимое файла gong.mat в рабочую область MATLAB:

```
clear
x=5;
whos
disp('Содержимое рабочей области до загрузки файла gong.mat:')
whos
disp('Содержимое файла gong.mat:')
```

```
whos('-file','gong.mat')
load('gong.mat')
disp('Содержимое рабочей области после загрузки файла gong.mat:')
whos
>> format1
  Name      Size            Bytes  Class     Attributes
  x         1x1                 8  double
Содержимое рабочей области до загрузки файла gong.mat:
  Name      Size            Bytes  Class     Attributes
  x         1x1                 8  double
Содержимое файла gong.mat:
  Name        Size            Bytes  Class     Attributes
  Fs          1x1                 8  double
  y       42028x1            336224  double
Содержимое рабочей области после загрузки файла gong.mat:
  Name        Size            Bytes  Class     Attributes
  Fs          1x1                 8  double
  x           1x1                 8  double
  y       42028x1            336224  double
```

Так как рабочая область определяет состояние системы в текущий момент времени, перед загрузкой данных из какого-либо файла лучше очистить ее содержимое от ранее загруженных значений переменных. Дело в том, что перезапись значений переменных данными из вновь считанного файла будет осуществлена, только если рабочая область уже содержит эти переменные. Во всех остальных случаях все ранее загруженные данные остаются в рабочей области, что может исказить результаты вычислений.

Очистить содержимое рабочей области можно командой **clear**. Например, очистим рабочую область MATLAB от значений переменных *y* Fs из файла gong.mat:

```
disp('Содержимое рабочей области до загрузки файла gong.mat:')
whos
disp('Содержимое файла gong.mat:')
whos('-file','gong.mat')
load('gong.mat')
disp('Содержимое рабочей области после загрузки файла gong.mat:')
whos
```

```
clear y Fs
disp('Содержимое рабочей области после удаления переменных y Fs')
whos
>> format1
Содержимое рабочей области до загрузки файла gong.mat:
  Name         Size            Bytes  Class     Attributes
  Fs           1x1                 8  double
  x            4x4               128  double
  y        42028x1            336224  double
Содержимое файла gong.mat:
  Name         Size            Bytes  Class     Attributes
  Fs           1x1                 8  double
  y        42028x1            336224  double
Содержимое рабочей области после загрузки файла gong.mat:
  Name         Size            Bytes  Class     Attributes
  Fs           1x1                 8  double
  x            4x4               128  double
  y        42028x1            336224  double
Содержимое рабочей области после удаления переменных y Fs
  Name      Size            Bytes  Class     Attributes
  x         4x4               128  double
```

Команда **clear**, записанная без аргументов, очищает всю рабочую область системы:

```
disp('Содержимое рабочей области до загрузки файла gong.mat:')
whos
disp('Содержимое файла gong.mat:')
whos('-file','gong.mat')
load('gong.mat')
disp('Содержимое рабочей области после загрузки файла gong.mat:')
whos
clear
disp('Содержимое рабочей области после очистки')
whos
>> format1
Содержимое рабочей области до загрузки файла gong.mat:
Fs           1x1                 8  double
  x            4x4               128  double
  y        42028x1            336224  double
```

Содержимое файла gong.mat:

```
  Name        Size              Bytes  Class     Attributes
  Fs          1x1                   8  double
  y       42028x1              336224  double
```

Содержимое рабочей области после загрузки файла gong.mat:

```
  Name        Size              Bytes  Class     Attributes
  Fs          1x1                   8  double
  y       42028x1              336224  double
```

Содержимое рабочей области после очистки

```
>>
```

ep

load filename – форма команды синтаксиса, позволяющая использовать меньшее количество специальных символов (круглых скобок, одинарных или двойных кавычек). Входные параметры разделяются пробелами вместо запятых. Например, следующие операторы, записывающие файл с именем durer.mat, эквивалентны:

```
load durer.mat
load ('durer.mat')
```

Можно включать любые из входных параметров, описанных в предыдущих синтаксисах. Например, чтобы загрузить переменную под названием *X*, используем операторы

```
load durer.mat
load('durer.mat', 'X')
```

7.1.2. MAT-файлы. Функция save

Функция **save** противоположна функции **load**. Она сохраняет переменные рабочей области в файл. Синтаксис следующий:

```
save (filename)
save (filename, variables)
save (filename, variables, fmt)
save (filename, variables, version)
save (filename, variables, version, '-nocompression')
save (filename, variables, '-append')
save (filename, variables, '-append', '-nocompression')
save filename
```

Функция **save (filename)** сохраняет все переменные из текущей рабочей области в отформатированный двоичный MAT-файл MATLAB

под названием filename. Если filename существует, функция **save** перезаписывает файл.

Функция **save (filename, variables, fmt)** сохраняет переменные рабочей области в файле, формат которого задан параметром fmt. Variables – дополнительный аргумент. Если аргумент variables не задан, функция **save** сохраняет все переменные рабочей области.

Функция **save (filename, variables, version)** сохраняет переменные в версию MAT-файла, заданную параметром version. Аргумент variables дополнительный.

Функция **save (filename, variables, version, '-nocompression')** сохраняет переменные в MAT-файл без сжатия. Опция '-nocompression' поддерживается только версией 7 MAT-файла, являясь в этой версии значением по умолчанию, поэтому для корректной работы опции необходимо задать значение version '-v7' или '-v7.3'.

Функция **save (filename, variables, '-append')** добавляет новые переменные к существующему MAT-файлу. Если переменная в файле уже существует, то функция **save** перезаписывает ее значение новым значением из рабочей области. Для ASCII-файлов опция '-append' добавляет данные в конец файла.

Функция **save (filename, variables, '-append', '-nocompression')** добавляет новые переменные к существующему файлу без сжатия. Версия существующего файла – 7 (значение по умолчанию) или 7.3.

save filename – форма команды синтаксиса, позволяющая использовать меньшее количество специальных символов (круглых скобок, одинарных или двойных кавычек). Входные параметры разделяются пробелами вместо запятых. Например, операторы, сохраняющие файл с именем test.mat, эквивалентны:

```
save test.mat
save ('test.mat')
```

Можно включать любые из входных параметров, описанных в предыдущих синтаксисах. Например, сохраним переменную под названием *X*:

```
save test.mat X       % command form
save('test.mat','X')  % function form
```

Не используйте форму команды, когда любые из входных параметров, таких как filename, – переменные или строки. Например:

Содержимое рабочей области до загрузки файла gong.mat:
Name Size Bytes Class Attributes
x 1x1 8 double
Содержимое файла gong.mat:
Name Size Bytes Class Attributes
Fs 1x1 8 double
y 42028x1 336224 double
Содержимое рабочей области после записи в файл gong.mat:
Name Size Bytes Class Attributes
x 1x1 8 double

Функция **save (filename, variables)** сохраняет только переменные или поля массива структур, заданного аргументом variables.

Для примера создадим массив *p* из 10 случайных чисел. Сохраним полученный массив в MAT-файле с именем pqfile.mat, затем просмотрим его содержимое.

```
p = rand (1, 10);
save ('pqfile.mat', 'p')
>>format1
p 1x10 80 double
```

Для примера сохраним массив *p* в ASCII-файл, а затем просмотрим содержимое этого файла с помощью функции **type**.

```
p = rand (1,10);
q = ones (10);
save ('pqfile.txt', 'p','q', '-ascii')
type ('pqfile.txt')
```

7.1.3. MAT-файлы. Функция matfile

С помощью этой функции в системе организован доступ к MAT-файлу и замена переменных в MAT-файле без загрузки файла в память. Синтаксис следующий:

```
matObj = matfile (filename)
matObj = matfile (filename, 'Writable', isWritable)
```

Команда **matObj = matfile (filename)** создает объект matlab.io.MatFile, соединенный с MAT-файлом, с именем filename. Объект MAT-файла позволяет получать доступ к переменным и заменять переменные непосредственно в MAT-файле без необходимости загружать переменные в память.

Команда **matObj = matfile (filename, 'Writable', isWritable)** включает или отключает доступ к файлу для записи. Если значение параметра isWritable равно true, то команда включает доступ к файлу для записи. В случае значения false – отключает доступ.

Входные параметры:

– filename – имя MAT-файла. Представляет собой полный или частичный путь к файлу;

– isWritable включает или отключает режим доступа для записи. Принимает одно из значений – true (для новых файлов) или false (для существующих файлов).

Как объект matObj обладает свойствами (функциями), доступ к которым осуществляется с помощью команды **matObj.Properties. имя свойства**. Перечень поддерживаемых системой свойств представлен ниже.

size	Возвращает измерения массива переменной в MAT-файле Команда **allDims = size (matObj, variable)** возвращает размер каждой размерности переменной variable, относящейся к объекту matObj Команда **[dim1, ..., dimN] = size (matObj, variable)** возвращает размеры каждой размерности переменной variable в отдельные выходные переменные dim1, ..., dimN Команда **selectedDim = size (matObj, variable, dim)** возвращает размер заданной размерности dim переменной variable
who	Возвращает список переменных в MAT-файле Команда **varlist = who (matObj)** возвращает в алфавитном порядке в массив ячеек varlist список всех переменных MAT-файла, относящихся к matObj Команда **varlist = who (matObj, variables)** перечисляет заданные переменные
whos	Выдает список переменных в MAT-файле с размерами и типами Команда **details = whos (matObj)** возвращает информацию обо всех переменных в MAT-файле, сопоставленном с matObj Команда **details = whos (matObj, VarName1, ..., VarNameN)** возвращает информацию о заданных переменных

В качестве примера создадим MAT-файл topography.mat и сохраним в нем переменную *a* рабочей области. Затем просмотрим содержимое файла.

```
m = matfile('topography.mat', 'Writable',true);
a = 12.7;
save('topography.mat', 'a');
whos('-file','topography.mat')
Name      Size            Bytes  Class     Attributes
a         1x1                 8  double
```

Помимо функций для импорта и экспорта данных, система MATLAB предлагает различные инструменты и режимы работы.

7.2. Импорт и экспорт данных с помощью инструментов MATLAB

MATLAB позволяет предварительно просматривать и считывать данные из текстовых файлов, электронных таблиц, оборудования, другого программного обеспечения или сети. Также можно импортировать изображения, видео- и аудиоформаты.

Рассмотрим различные способы ввода данных в MATLAB, связанные с режимами работы системы. Данные находятся в различных источниках.

Воспользоваться инструментом *Import Tool* можно, перейдя на вкладку *Home*, в раздел *Variable*, выбрав инструмент *Import Data* (рис. 7.1).

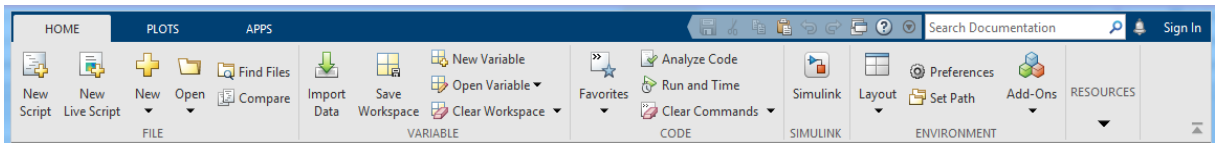


Рис. 7.1. Раздел *Variable* вкладки *Home*

В качестве примера импортируем с помощью инструмента *Import Tool* xls-файл «Успеваемость». Для этого выполним следующую последовательность действий: выберем инструмент *Import Data*, пройдем в рабочий каталог, где хранится файл «Успеваемость.xlsx», выберем указанный файл. Импортированный файл откроется в отдельном окне *Import* (рис. 7.2).

В качестве альтернативы в браузере текущей папки дважды кликните имя файла с расширением .xls, .xlsx, .xlsb или .xlsm. Инструмент *Import Tool* откроется автоматически.

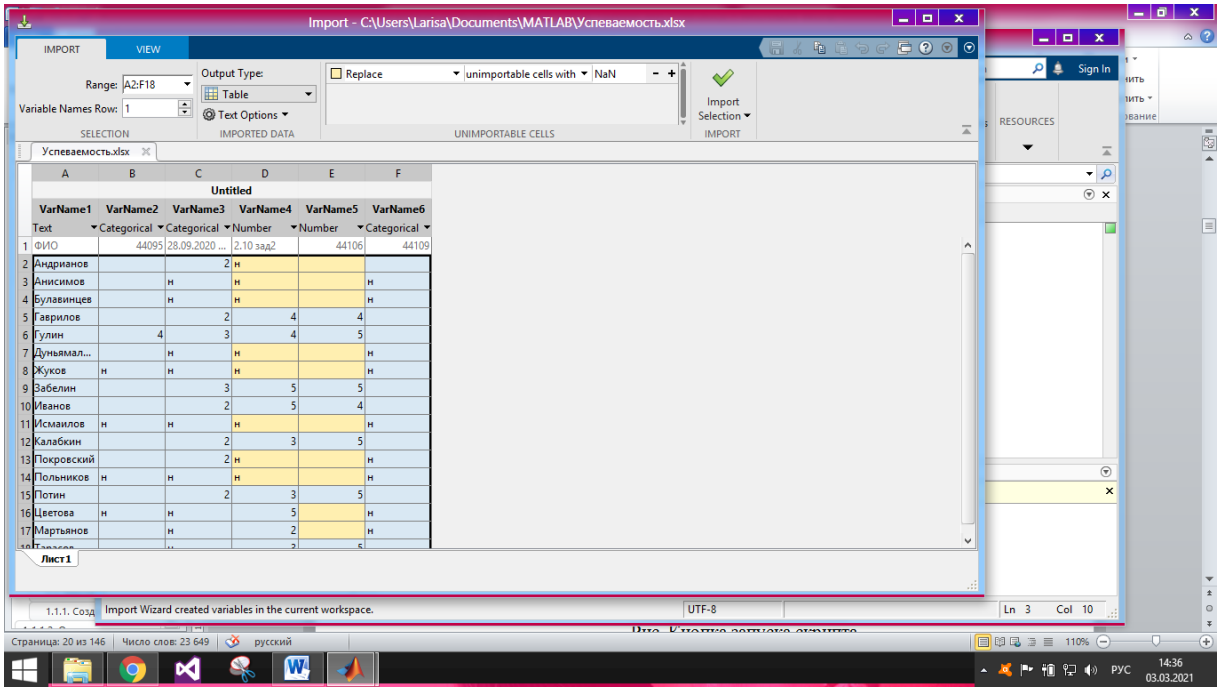


Рис. 7.2. Внешний вид импортированного файла

На вкладке *Import*, в разделе *Output Type* можно задать, как импортировать данные (рис. 7.3). Выберем тип *Table*.

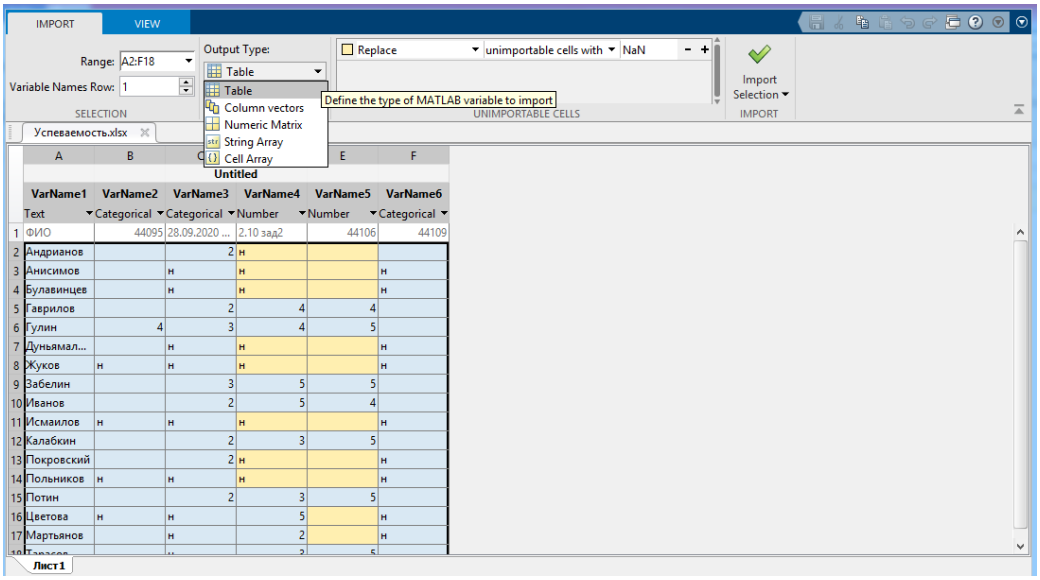


Рис. 7.3. Раздел *Output Type* вкладки *Import*

Основные опции перечислены ниже. Выбранная опция диктует тип импортируемых данных.

Вектор-столбец	Импортирует каждый столбец выбранных данных как отдельный m – 1 вектор
Числовая матрица	Импортирует выбранные данные как m – n числовой массив
Массив строк	Импортирует выбранные данные как m – n массив строк
CellArray	Импортирует выбранные данные как массив ячеек, который может содержать несколько типов данных, таких как числовые данные и текст
Таблица	Импортирует выбранные данные как таблицу

Если пользователь принимает решение импортировать данные как матрицу или числовые вектор-столбцы, инструмент *Import Tool* подсвечивает любые нечисловые данные в рабочем листе. Каждый цвет подсветки соответствует предложенному правилу превратить совпадение данных в числовой массив. Например, можно заменить нечисловые значения на NaN. Кроме того, если установить курсор на отдельные ячейки, можно увидеть, как данные будут импортированы (рис. 7.4).

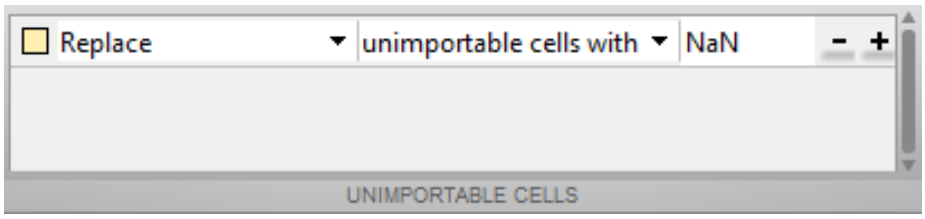


Рис. 7.4. Внешний вид *Import Tool*

Можно добавить, удалить, переупорядочить или отредактировать правила, например изменение заменяющего значения от NaN к другому значению. Все правила применяются только к импортированным данным и не изменяют данные в файле.

Любые ячейки, которые содержат значение ячейки #Error?, соответствуют ошибкам формулы в исходном файле электронной таблицы (напрмер, деление на ноль). Инструмент *Import Tool* рассматривает эти ячейки как нечисловые (рис. 7.5).

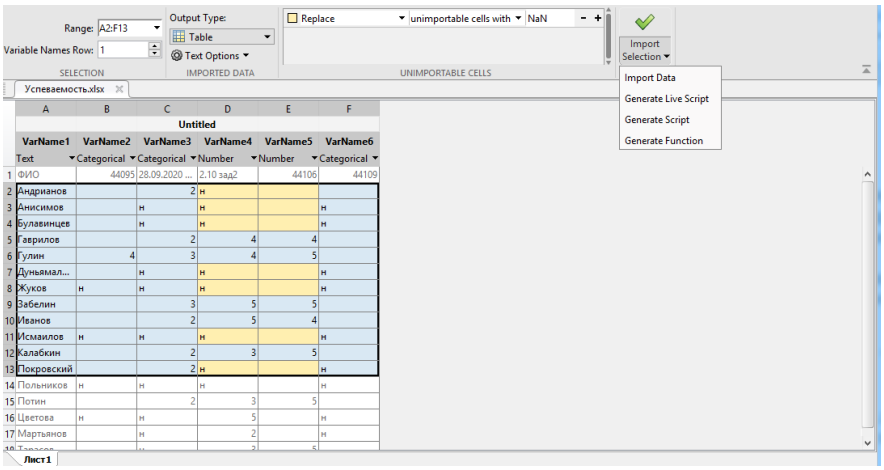


Рис. 7.5. Внешний вид нечисловых значений

Можно импортировать всю таблицу целиком, а можно выбрать данные для импорта.

Раздел *Import Selection* позволяет сохранить данные либо в переменной рабочей области, либо в виде скрипта или функции.

В качестве примера импортируем данные о студентах группы в переменную *T*. Для этого выполним следующее.

На вкладке *Import Selection* выберем инструмент *Import Data*. Данные сохранятся в переменной *Untitled*, которую в дальнейшем переименуем в переменную *T* во вкладке *Variable* (рис. 7.6).

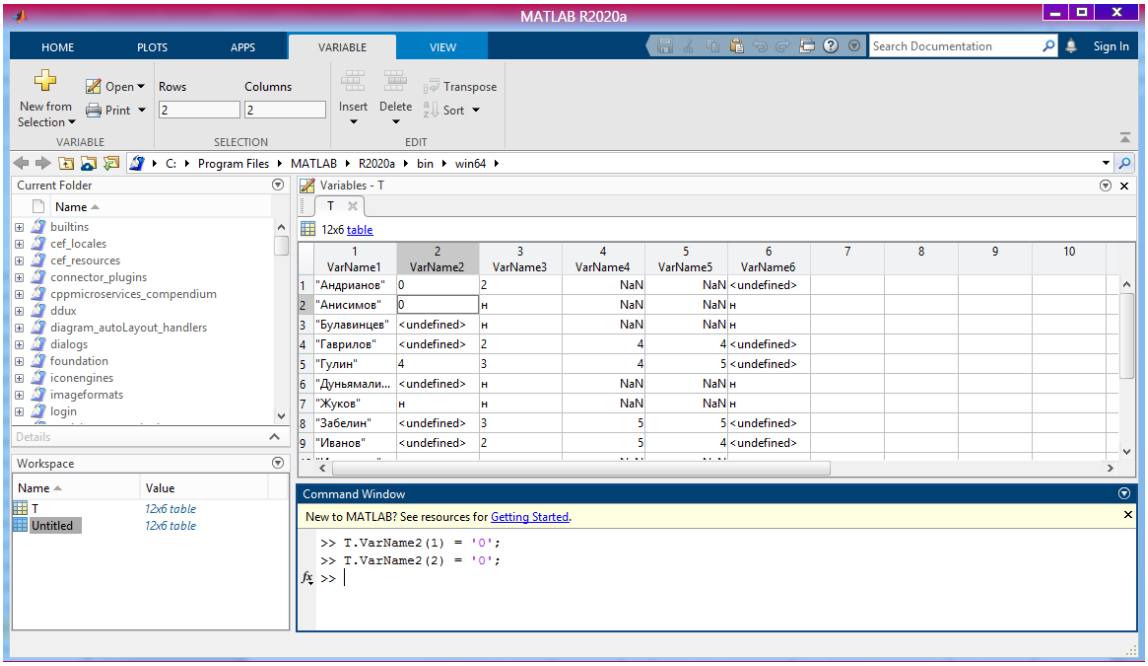


Рис. 7.6. Внешний вид вкладки *Import Selection*

Как видно по рис. 7.6, в исходных данных много пропусков, названия полей заданы некорректно. Для дальнейшей работы можно отредактировать и сохранить данные.

Заменим названия полей, введем недостающие оценки. Результат работы представлен ниже.

```
>> summary (T)
Variables:
Фамилия: 17×1 string
2_10: 17×1 categorical
Values:
н        0
4       13
3        2
2        2
7_10: 17×1 categorical
Values:
н       0
2       8
3       5
5       4
12_10: 17×1 double
Values:
Min         2
Median      4
Max         5
17_10: 17×1 double
Values:
Min         3
Median      4
Max         5
22_10: 17×1 categorical
Values:
н        2
3       12
2        2
5        1
```

Сохраним значения переменной *T* рабочей области в MAT-файле instrument2.mat. Для этого выберем на вкладке *Variable* раздел *Save Workspace*.

Посмотрим общую информацию о таблице с помощью функции **summary**, выводящей общую статистику о каждом поле таблицы. Как видно из примера ниже, поле «Фамилия», например, содержит 17 строковых данных.

```
>> summary (T)
Variables:
   Фамилия: 17×1 string
   2_10: 17×1 categorical
      Values:
         н              4
         4              1
         NumMissing     12
   7_10: 17×1 categorical
      Values:
         н      9
         2      6
         3      2
   12_10: 17×1 double
      Values:
         Min            2
         Median         4
         Max            5
         NumMissing     8
   17_10: 17×1 double
      Values:
         Min            4
         Median         5
         Max            5
         NumMissing     10
   22_10: 17×1 categorical
      Values:
         н              9
         NumMissing     8
```

Экспортируем таблицу во внешний файл otchet.xls, находящийся на диске *C*. Для этого предварительно создадим этот файл в указанном каталоге. Откроем файл instrument2.mat и в режиме командной строки выполним команду

```
>> save ('C:\ot.xls','T')
```

С помощью инструмента *Import Tool* можно импортировать данные из нескольких электронных таблиц. Например, предположим, что есть набор электронных таблиц в текущей папке с названиями myfile01.xlsx, myfile25.xlsx и необходимо импортировать одну и ту же область значений данных – A2:G100 – из первого рабочего листа в каждом файле. Сгенерируем код для импорта всего набора файлов следующим образом:

1) откроем один из файлов, например «Успеваемость.xlsx», в *Import Tool*;

2) во вкладке *Import Selection* выберем инструмент *Generate Function*. Инструмент *Import Tool* сгенерирует код, похожий на следующую выборку, и откроет код в *Editor*:

```
%% Input handling
 % If no sheet is specified, read first sheet
if nargin == 1 || isempty(sheetName)
   sheetName = 1;
end
% If row start and end points are not specified, define defaults
if nargin <= 2
   dataLines = [2, 18];
end
%% Setup the Import Options and import the data
opts = spreadsheetImportOptions("NumVariables", 6);
% Specify sheet and range
opts.Sheet = sheetName;
opts.DataRange = "A" + dataLines(1, 1) + ":F" + dataLines(1, 2);
% Specify column names and types
opts.VariableNames = ["VarName1", "VarName2", "VarName3", "VarName4", "VarName5", "VarName6"];
opts.VariableTypes = ["string", "categorical", "categorical", "double", "double", "categorical"];
% Specify variable properties
```

```
opts = setvaropts(opts, "VarName1", "WhitespaceRule", "preserve");
opts = setvaropts(opts, ["VarName1", "VarName2", "VarName3", "VarName6"], "EmptyFieldRule", "auto");
% Import the data
Untitled = readtable(workbookFile, opts, "UseExcel", false);
for idx = 2:size(dataLines, 1)
    opts.DataRange = "A" + dataLines(idx, 1) + ":F" + dataLines(idx, 2);
    tb = readtable(workbookFile, opts, "UseExcel", false);
    Untitled = [Untitled; tb]; %#ok<AGROW>
end
end
```

3) сохраним функцию в виде скрипта, например под именем instrument3.m;

4) в отдельном программном файле или командной строке создадим цикл for, чтобы импортировать данные из каждой электронной таблицы в массив ячеек под названием myData:

```
numFiles = 25;
range = 'A2:G100';
sheet = 1;
myData = cell(1,numFiles);
for fileNum = 1:numFiles
    fileName = sprintf('myfile%02d.xlsx',fileNum);
    myData{fileNum} = importfile(fileName,sheet,range);
end
```

Каждая ячейка в массиве ячеек myData содержит массив данных из соответствующего рабочего листа. Например, myData{1} содержит данные из первого файла myfile01.xlsx.

7.3. Импорт и экспорт данных в программном режиме и режиме командной строки

В режиме командной строки файл импортируется с помощью команды **uiimport (filename)**. Для импорта файла «Успеваемость.xlsx» в командной строке наберем команду

```
uiimport ('C:\Успеваемость.xlsx')
```

и нажмем клавишу *Enter*.

Для импорта данных в программном режиме нам понадобится функция **importdata ()**, позволяющая загружать файлы данных различных форматов. Синтаксис следующий:

A = importdata (filename)
A = importdata ('-pastespecial')
A = importdata (___, delimiterIn)
A = importdata (___, delimiterIn, headerlinesIn)
[A, delimiterOut, headerlinesOut] = importdata (___)

Команда **A = importdata (filename)** загружает данные в массив *A* из файла, обозначенного filename.

Команда **A = importdata (___, delimiterIn)** интерпретирует delimiterIn как разделитель столбцов в файле ASCII, файле с именем filename или для данных из буфера обмена. Можно использовать delimiterIn с любым из входных аргументов в приведенных выше синтаксисах.

Команда **A = importdata (___, delimiterIn, headerlinesIn)** загружает данные из файла ASCII, файла с именем filename или буфера обмена, считывая числовые данные, начиная со строки headerlinesIn + 1.

Команда **[A, delimiterOut, headerlinesOut] = importdata (___)** возвращает обнаруженный символ разделителя для входного файла ASCII в delimiterOut и обнаруженное количество строк заголовка в headerlinesOut, используя любой из входных аргументов в предыдущих синтаксисах.

Пример. Импортируем в MATLAB текстовый файл weekdata.txt, находящийся на съемном диске *D*. Для этого выполним следующую последовательность действий.

1. Используя *Блокнот*, создадим разграниченный пробелом ASCII-файл под названием weekdata.txt с заголовками столбцов Day1, Day2, …, Day7.

Внешний вид файла представлен на рис. 7.7.

```
Day1  Day2  Day3  Day4  Day5  Day6  Day7
95.01 76.21 61.54 40.57  5.79 20.28  1.53
23.11 45.65 79.19 93.55 35.29 19.87 74.68
60.68  1.85 92.18 91.69 81.32 60.38 44.51
48.60 82.14 73.82 41.03  0.99 27.22 93.18
89.13 44.47 17.63 89.36 13.89 19.88 46.60
```

Рис. 7.7. Внешний вид текстового файла

2. Создадим файл сценария week_data.m, содержащий следующий код. Зададим разделитель в виде пробела и заголовок отдельного столбца:

```
filename = 'D:\weekdata.txt';
delimiterIn = ' ';
headerlinesIn = 1;
A = importdata (filename, delimiterIn, headerlinesIn);
```

3. Просмотрим 3-й и 5-й столбцы:

```
for k = [3, 5]
disp (A.colheaders{1, k})
disp (A.data (:, k))
disp (' ')
end
```

Результат работы скрипта:

```
Day3
   61.5400
   79.1900
   92.1800
   73.8200
   17.6300
Day5
    5.7900
   35.2900
   81.3200
    0.9900
   13.8900
```

Сохраним текстовый файл в текущем каталоге проекта.

7.4. Импорт и экспорт изображений, видео- и аудиофайлов в MATLAB

Первые два рассмотренных способа не импортируют изображения, видео- и аудиоформаты.

В качестве примера импортируем и отобразим файл изображения, находящийся в папке img на локальном диске *C*. Для этого выполним следующую последовательность действий:

1) создадим файл сценария img_scr.m, содержащий следующий код:

Рис. 7.8. Кнопка запуска скрипта

```
filename ='C:\img\logo.jpg';
A = importdata (filename);
image(A);
```

2) запустим скрипт командой *Run* вкладки *Editor* на панели инструментов или с помощью клавиши F5 на клавиатуре (рис. 7.8);

3) MATLAB отобразит файл изображения в отдельном окне *Figure* (рис. 7.9);

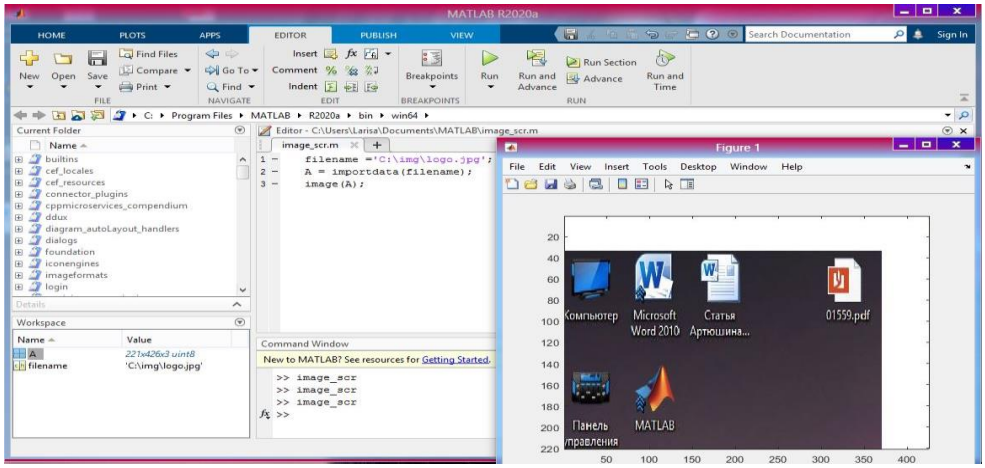


Рис. 7.9. Отображение графических файлов в системе MATLAB

4) сохраним файл изображения в текущем каталоге проекта, например так: ‘Рабочий стол/data/img_logo/logo.fig’

7.5. Импорт и экспорт данных из буфера обмена

Система MATLAB позволяет загружать данные не только из файла, но и из буфера обмена. Для этого в функции **importdata** вместо имени файла указывается ключевое слово ‘-pastespecial’:

```
A = importdata (‘-pastespecial’);
```

Пример. Импортируем в MATLAB данные буфера обмена. Для этого выполним следующую последовательность действий:

1) наберем в текстовом редакторе *Блокнот* следующие строки:

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

2) скопируем строки в буфер обмена. Выберем текст, щелкнем правой кнопкой мыши и затем выберем команду *Copy*;

3) импортируем данные из буфера обмена в MATLAB путем ввода следующей команды: A = importdata ('-pastespecial').

Результат работы представлен на рис. 7.10.

```
>> A = importdata('-pastespecial')

A =

     1     2     3
     4     5     6
     7     8     9

>>
```

Рис. 7.10. Импорт данных из буфера обмена

Самый простой способ сохранить (экспортировать) значения всех переменных – использовать в меню *File* пункт *Save Workspace As*. Сохраним значение переменной *A* в MAT-файле matlab1.mat.

7.6. Основные функции для считывания и записи данных в таблицу

Для считывания и записи данных из файла в файл (в таблицу/из таблицы) используются следующие функции.

readtable	Составление таблицы из файла
writetable	Запись таблицы в файл
detectImportOptions	Создание настроек импорта на основе содержимого файла
spreadsheetImportOptions	Объект, управляющий процессом импорта табличных данных из файлов электронных таблиц
getvaropts	Получение переменных настроек импорта
setvaropts	Установка переменных настроек импорта
setvartype	Установка типа данных переменных
preview	Предварительный просмотр восьми строк из файла с помощью параметров импорта

Более подробную информацию об импорте и экспорте данных можно получить, воспользовавшись справочной системой MATLAB или документацией на сайте exponenta.ru.

Вопросы для самоконтроля

1. Приведите примеры классификации данных по критериям, отличным от приведенных в тексте главы.

2. Назовите основные виды обработки данных. Охарактеризуйте каждый вид.

3. Продемонстрируйте на конкретном примере, как импортировать и экспортировать текстовые или числовые данные с помощью специальных функций.

4. Продемонстрируйте на конкретном примере, как импортировать и экспортировать текстовые или числовые данные с помощью инструмента *Import Tool*.

5. Продемонстрируйте на конкретном примере, как импортировать и экспортировать текстовые или числовые данные в режиме командной строки.

6. Продемонстрируйте на конкретном примере, как импортировать и экспортировать текстовые или числовые данные из буфера обмена.

7. Продемонстрируйте на конкретном примере, как импортировать и экспортировать изображения, аудио- или видеоформаты (на выбор).

8. Расскажите, какие возможности предоставляют форматы table и typetable.

Практическая работа

ИМПОРТ ДАННЫХ ИЗ ИНТЕРНЕТ-ИСТОЧНИКА В MATLAB. ПРЕОБРАЗОВАНИЕ ДАННЫХ

Задание. Импортируйте данные из интернет-источника, указанного ниже. Преобразуйте и проанализируйте данные согласно требованиям, указанным в задании. Оформите отчет о проделанной работе, содержащий:

– титульный лист (см. приложение);

– текст задания;

– тексты скриптов и скриншоты результатов выполнения каждого пункта задания.

I. Импортируем в MATLAB данные статистики числа зараженных COVID19 в некоторых странах мира. Для этого последовательно выполним следующие действия.

1. Данные в MATLAB в виде xlsx-файла загрузим с сайта https://ourworldindata.org/coronavirus-source-data с помощью скриптов проекта fitVirusCOVID19 с сайта https://www.mathworks.com/matlabcentral/fileexchange/74658 (рис. 7.11).

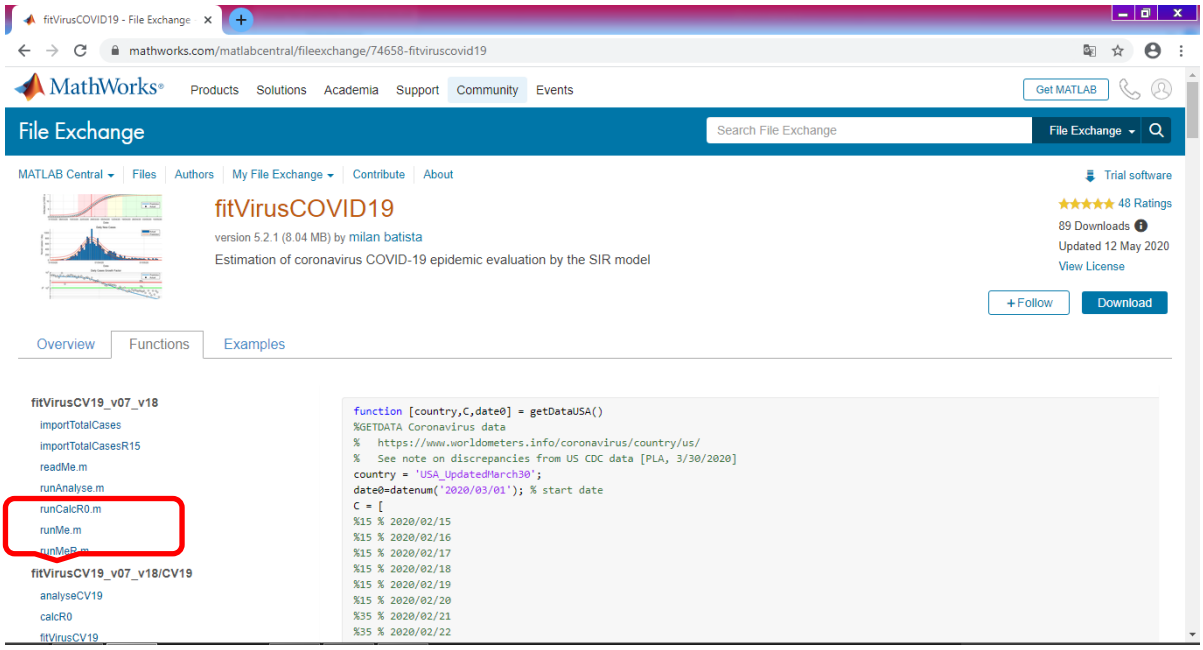


Рис. 7.11. Команды скрипта проекта fitVirusCOVID19

2. Создадим в MATLAB новый скрипт с именем importTotal Cases.m. Скопируем в него содержимое скрипта проекта fitVirus COVID19. Запустим его на выполнение.

3. В результате работы скрипта с сайта ourworldindata.org будет скачан файл totalcases20200917.xlsx, содержащий данные о числе зараженных COVID19 в некоторых странах мира (рис. 7.12).

4. Откроем скачанный файл в MATLAB.

II. Преобразуем данные в табличный формат.

1. Считаем данные в переменную *T*. Первый столбец – дата, остальные столбцы – названия стран, значения ячеек – количество заболевших в каждой стране на конкретную дату (рис. 7.13).

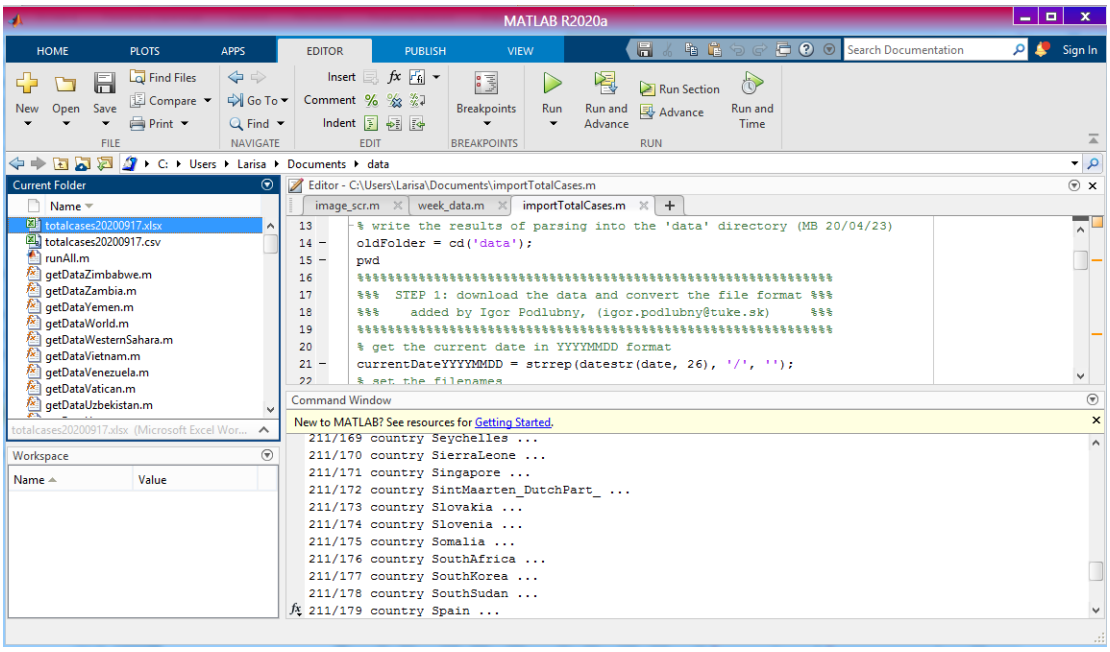


Рис. 7.12. Содержимое файла totalcases20200917.xlsx

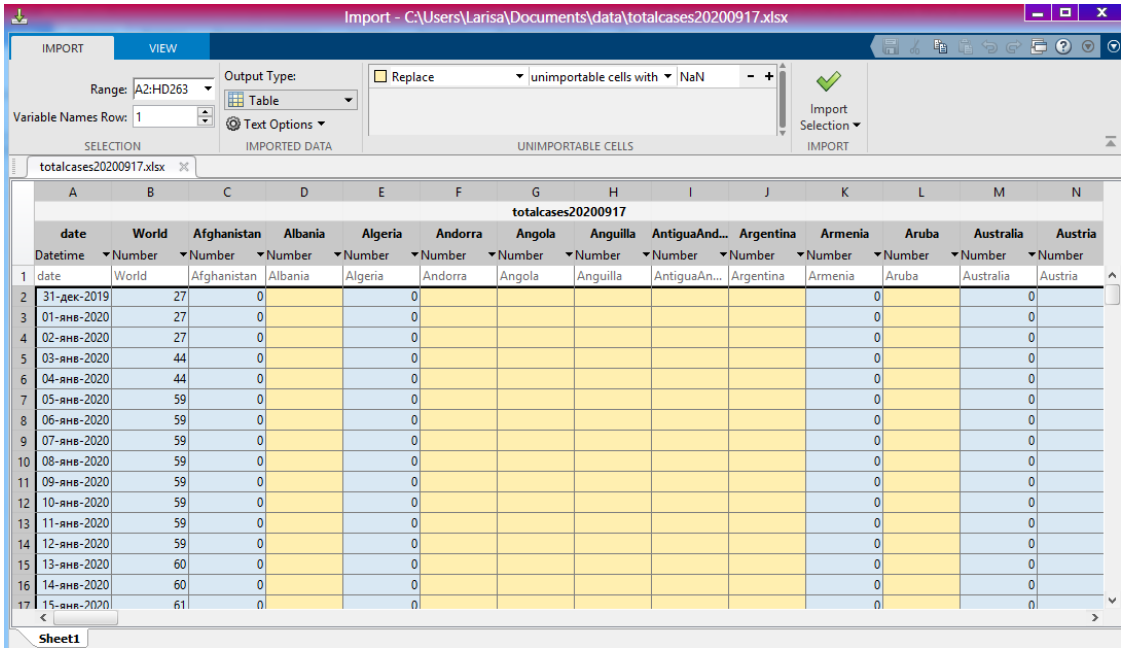


Рис. 7.13. Файл с данными для загрузки в MATLAB

2. Для более удобной работы с данными типа «дата» переведем данные в табличный формат.

3. Сохраним в отдельную таблицу (T1) данные по следующим странам: Китай, Италия, Россия, Швеция. Визуализируем таблицу.

4. Файлы скрипта и таблицы, оформленный отчет необходимо предоставить преподавателю.

Глава 8
ПРЕДОБРАБОТКА ИМПОРТИРОВАННЫХ ДАННЫХ

8.1. Предобработка данных с помощью инструмента *Live Editor*

Предварительная обработка данных – важный шаг в процессе интеллектуального анализа данных. Фраза «мусор на входе – мусор на выходе» [1] применима, в частности, и для проектов интеллектуального анализа данных. Здесь имеется в виду то, что даже самый изощренный анализ не принесет пользы, если за основу взяты сомнительные данные.

Почему это происходит? Методы сбора данных часто плохо контролируются. Это приводит к появлению недопустимых значений (например, доход, равный −100), комбинаций данных, которые невозможны (например, «мужской пол при наличии беременности»), отсутствию значений и пр. В результате анализа данных, которые не защищены от такого рода проблем, можно прийти к неверным выводам, поэтому контроль качества импортируемых данных – первостепенная задача при проведении анализа.

Предобработку можно выполнять автоматически в соответствии с определенным набором правил или вручную. Предобработка данных включает в себя:

– очистку – процесс выявления и исправления несоответствия данных с целью улучшения их качества;

– нормализацию – приведение данных к простейшему либо каноническому виду с помощью эквивалентных преобразований. Используется для стандартизации диапазона значений независимых переменных или признаков данных (например, сведение к интервалам [0; 1] или [–1; +1]);

– выделение признаков – это процесс снижения размерности данных, в котором исходный набор сокращается до более управляемых групп, оставаясь при этом достаточным для точного и полного описания исходного набора данных.

В системе MATLAB за предобработку данных отвечает инструмент *Live Editor* («Живые задачи», рис. 8.1, 8.2, 8.3). Расположен на вкладке *Insert*, меню *Task*. Представляет собой набор типовых задач предобработки данных (Data Preprocessing), преобразования таблиц (Tables and Timetables) и проектирования контроллеров для систем с обратной связью (Control System Designer). В рамках пособия рассматриваются два первых процесса.

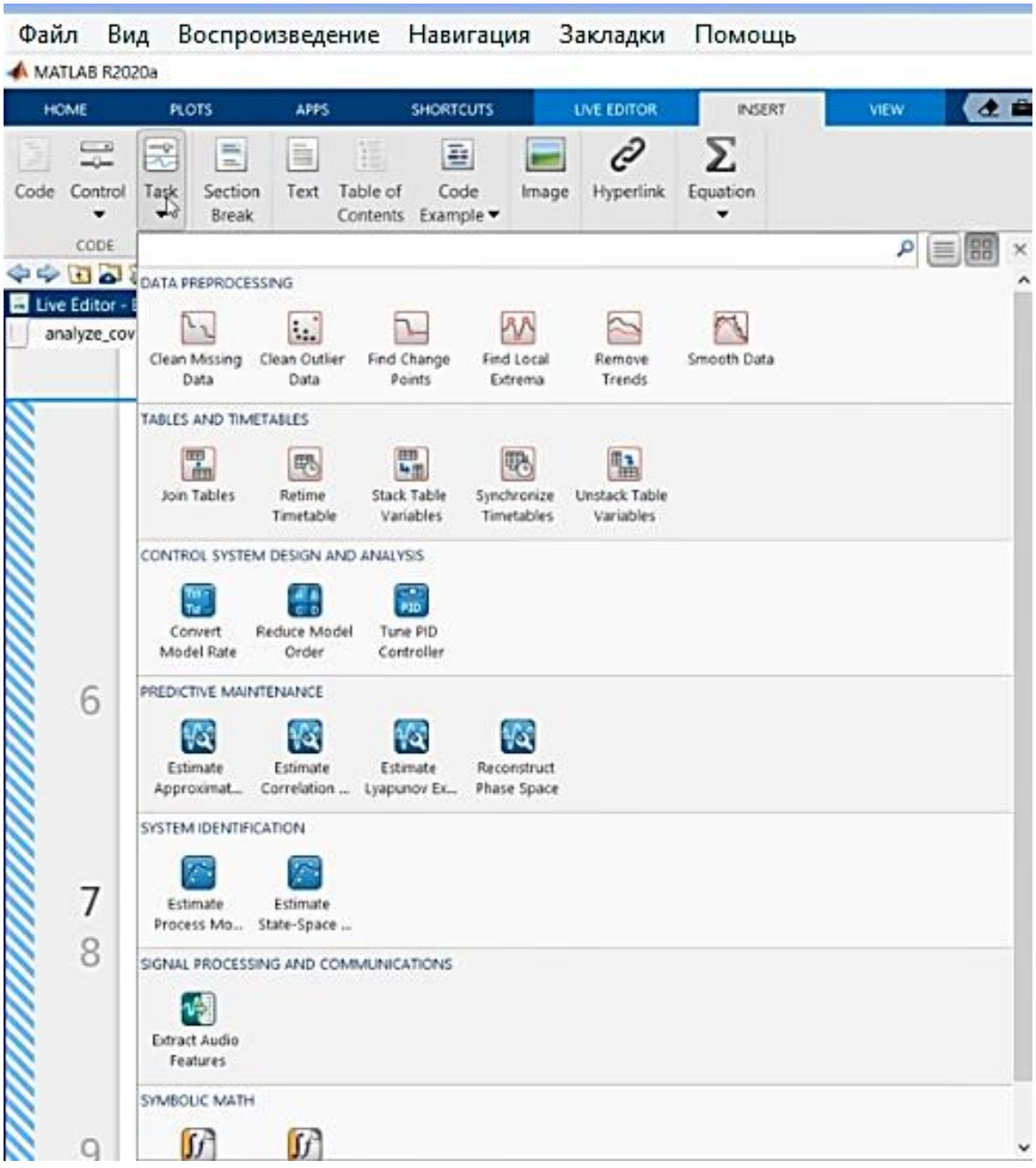


Рис. 8.1. Инструмент *Live Editor*

Рис. 8.2. Окно вкладки предобработки данных

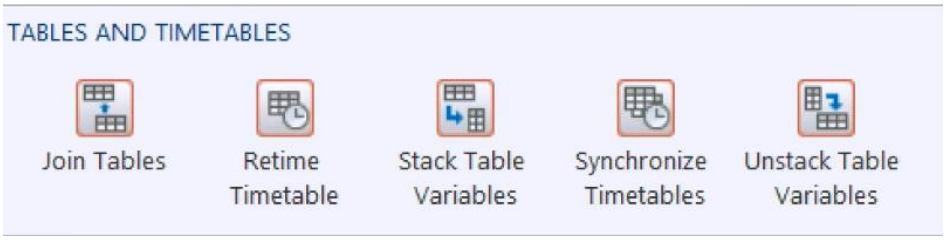


Рис. 8.3. Окно вкладки преобразования таблиц

В версии MATLAB R2020a реализованы следующие типовые задачи предобработки данных:

– Clean Missing Data – задача, позволяющая в интерактивном режиме обработать недостающие значения данных, такие как NaN (для числовых данных) или <missing> (для других типов данных). Задача автоматически генерирует код MATLAB для live-скрипта. Используя эту задачу, можно найти, заполнить или удалить недостающие данные в переменной рабочей области; настроить метод для заполнения данных; автоматически визуализировать недостающие и удаленные данные;

– Clean Outlier Data – задача, позволяющая в интерактивном режиме обработать выбросы в данных. Автоматически генерирует код MATLAB для live-скрипта. Используя эту задачу, можно найти, заполнить или удалить выбросы из данных в переменной рабочей области; настроить методы для нахождения и заполнения выбросов; автоматически визуализировать данные о выбросе и удаленные данные;

– Find Change Points – поиск точек экстремума. Позволяет в интерактивном режиме найти резкие изменения в среднем значении, отклонении или наклоне и прерывании данных. Задача автоматически

генерирует код MATLAB для live-скрипта. Используя эту задачу, можно найти точки перехода в данных переменной рабочей области; настроить количество обнаруживаемых точек перехода; автоматически визуализировать местоположение точки перехода и сегментов данных между ними;

– Find Local Extrema – поиск локальных минимумов и максимумов. Задача автоматически генерирует код MATLAB для live-скрипта. Позволяет в интерактивном режиме находить и автоматически визуализировать локальные максимумы и минимумы в данных переменной рабочей области;

– Remove Trends – удаление полиномиального тренда из данных. Задача автоматически генерирует код MATLAB для live-скрипта. Используя эту задачу, можно выбрать степень полиномиального тренда, чтобы удалить из данных в переменной рабочей области; динамически расположить точки останова, чтобы задать кусочные сегменты данных; задать ограничения непрерывности; автоматически визуализировать вычисленный тренд и данные с удаленным трендом;

– Smooth Data – сглаживание зашумленных данных. Задача автоматически генерирует код MATLAB для live-скрипта. Используя эту задачу, можно настроить метод для сглаживания данных в переменной рабочей области и параметры, чтобы сгенерировать степень сглаживания; автоматически визуализировать сглаженные данные.

В качестве примера создадим и построим вектор из «грязных» данных, которые содержат четыре NaN значения и пять выбросов:

```
x = 1:100;
data = cos(2*pi*0.05*x+2*pi*rand) + 0.5*randn(1,100);
data(20:20:80) = NaN;
data(10:20:90) = [-50 40 30 -45 35];
```

Визуализируем данные. Сделать это можно, открыв задачу *Создать* во вкладке *Plot* (рис. 8.4). На графике отчетливо видны точки разрыва функции и выбросы.

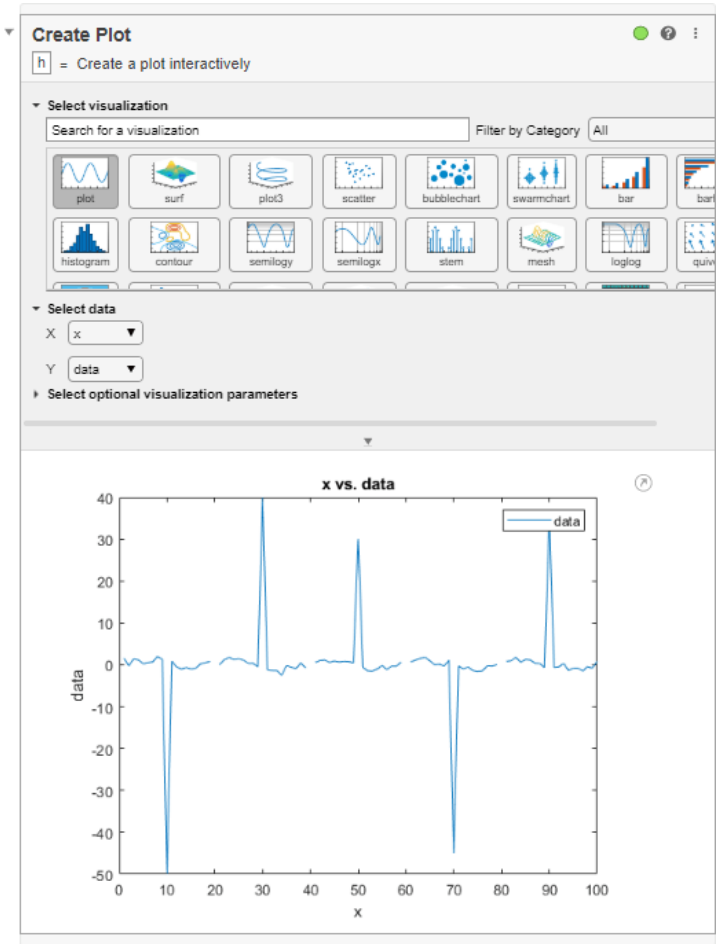


Рис. 8.4. Визуализация данных инструментом *Plot*

Предобработку данных выполним в четыре этапа.

На первом этапе найдем и заполним недостающие данные с помощью инструмента *Clean Missing Data* (рис. 8.5), выбрав в качестве исходных данных значения переменной data, а в качестве метода очистки – Lianer Interpolation (рис. 8.6).

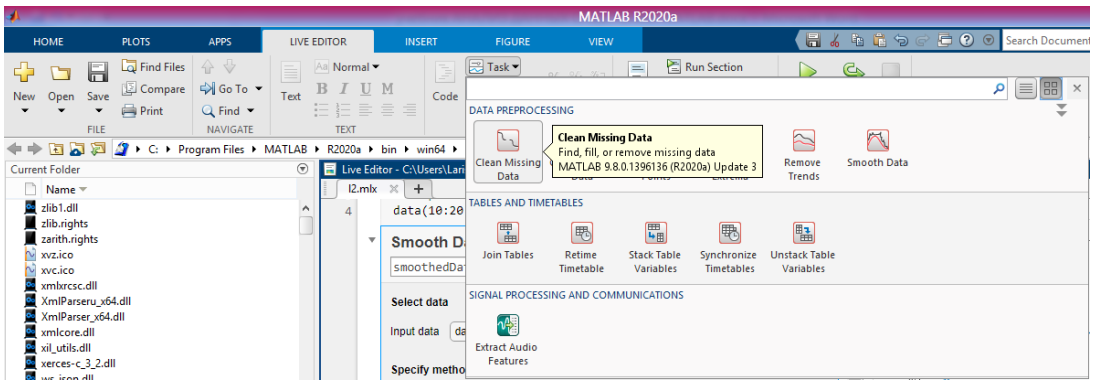


Рис. 8.5. Выбор инструмента *Clean Missing Data*

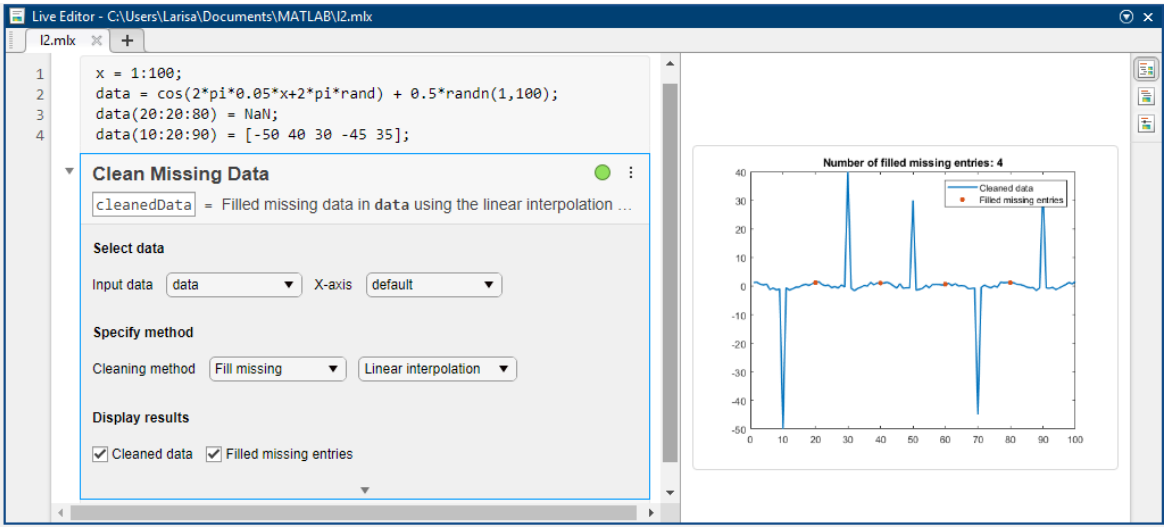


Рис. 8.6. Данные после заполнения значений NaN

На втором этапе удалим выбросы с помощью инструмента *Clean Outlier Data* (рис. 8.7), выбрав в качестве входных данных результаты работы инструмента *Clean Missing Data* – cleanedData.

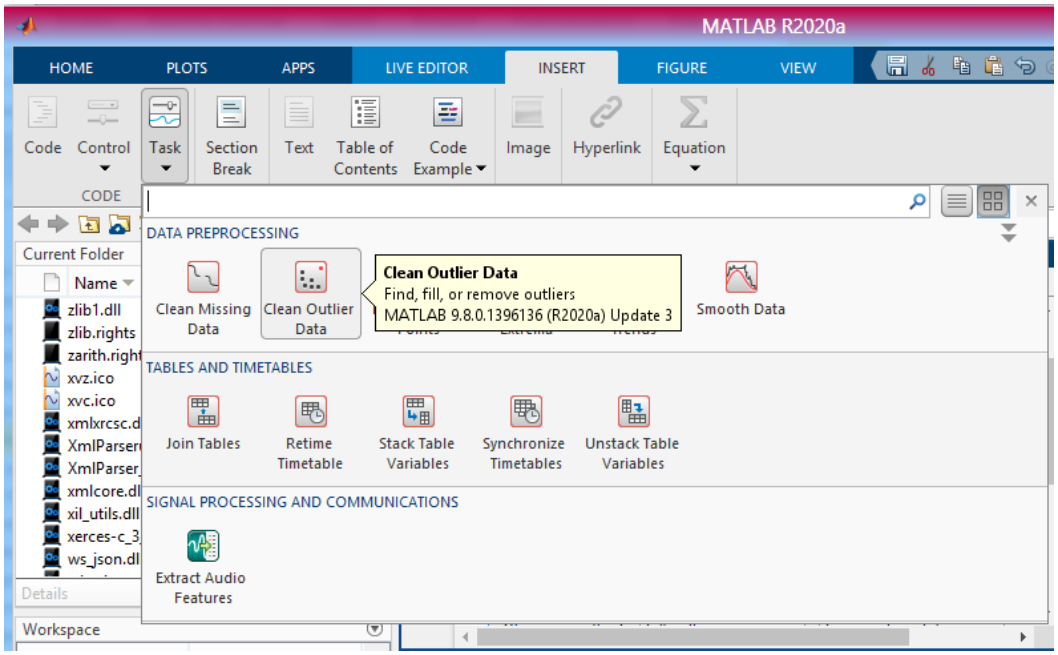


Рис. 8.7. Запуск задачи *Clean Outlier Data*

На третьем этапе сгладим убранные данные из предыдущей задачи при помощи *Smooth Data* (рис. 8.8), выбрав в качестве входных данных результаты работы инструмента *Clean Outlier Data* – cleanedData2.

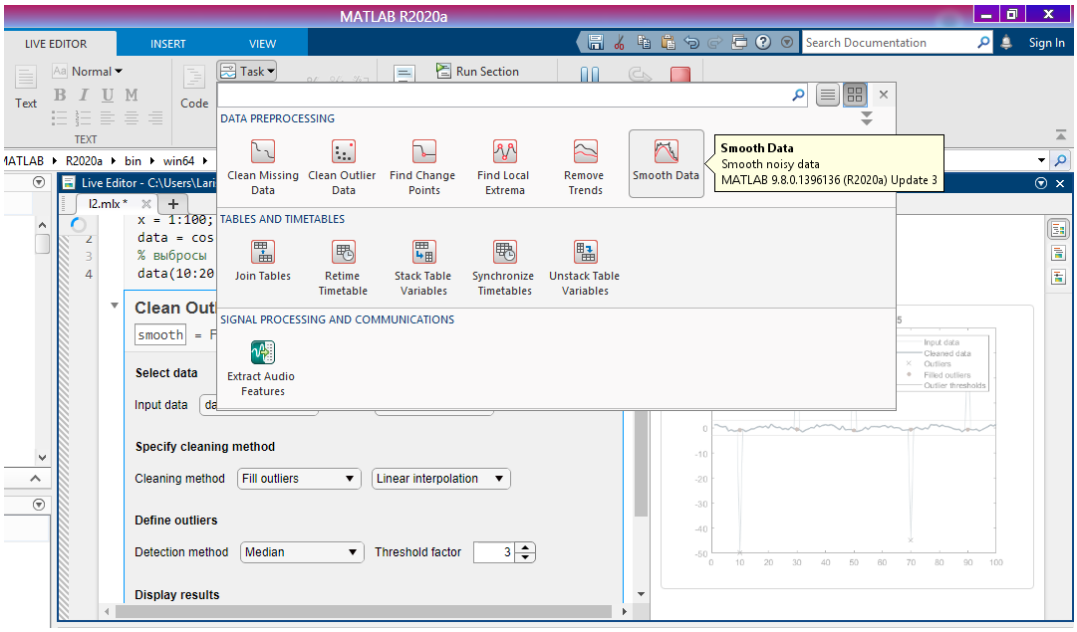


Рис. 8.8. Выбор инструмента *Smooth Data*

На четвертом этапе для поиска и замены значения NaN в данных воспользуемся инструментом *Clean Missing Data*. Для этого на вкладке *Live Editor* последовательно выберем *Task*, *Clean Missing Data* (см. рис. 8.5). Выберем входные данные (data) и метод очистки (Linear Interpolation), чтобы отобразить заполненные данные на графике автоматически.

Для поиска и удаления выбросов запустим задачу *Clean Outlier Data* (рис. 8.9, 8.10).

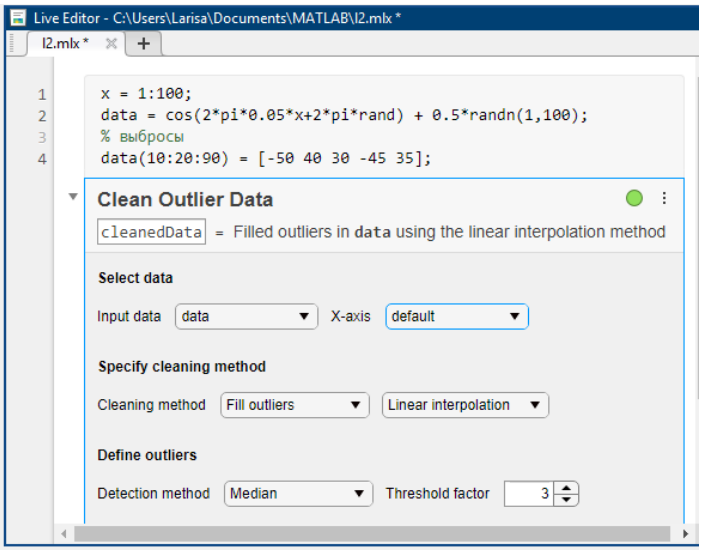


Рис. 8.9. Окно входных данных
для визуализации инструментом *Live Editor*

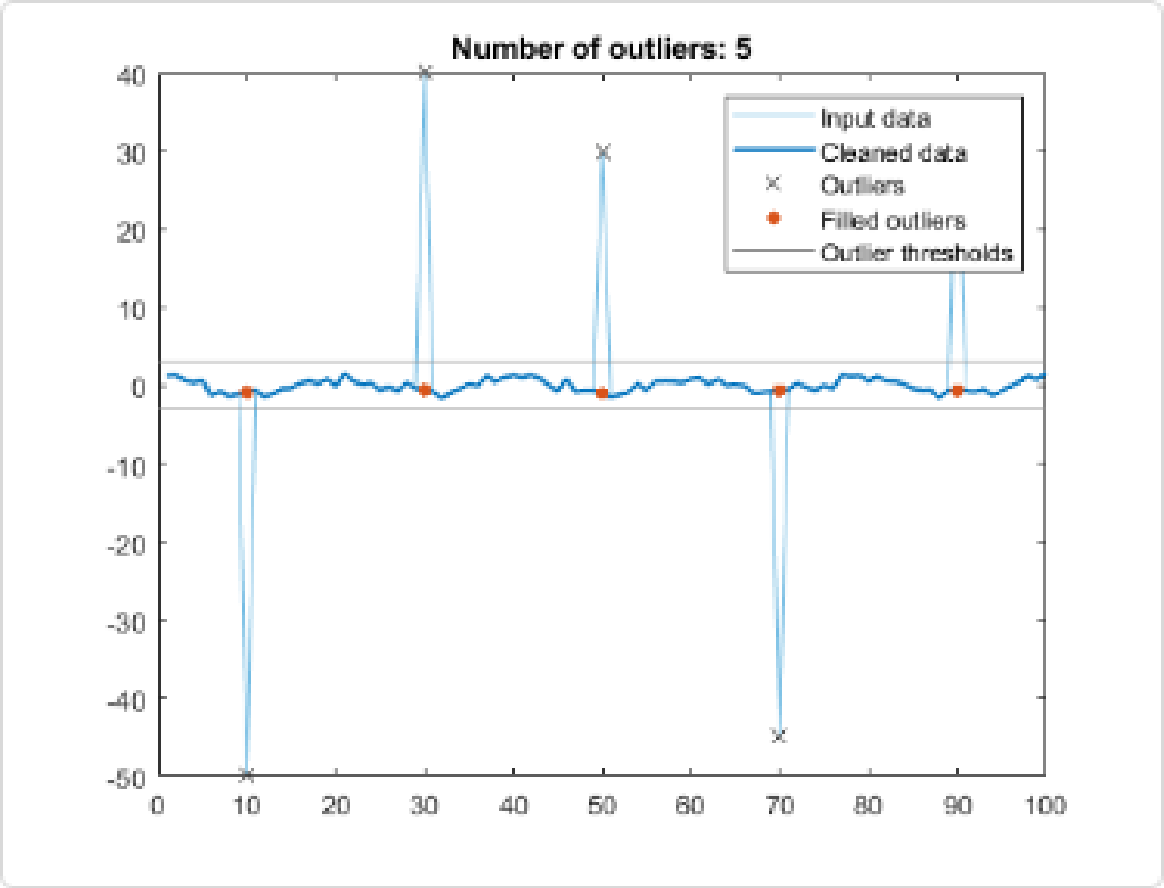


Рис. 8.10. Визуализация выбросов
инструментом *Clean Outlier Data*

На графике (см. рис. 8.10) явно видны координаты выбросов, также инструмент показывает их количество: «Number of outliers: 5».

Сгладим выбросы при помощи инструмента *Smooth Data*. Для этого на вкладке *Live Editor* последовательно выберем *Task, Smooth Data* (см. рис. 8.8).

В открывшемся диалоговом окне инструмента выберем переменную, значения которой необходимо сгладить. В нашем случае это переменная data (рис. 8.11). Результат сглаживания представлен на рис. 8.12, 8.13. Данные подготовлены для последующего анализа и обработки.

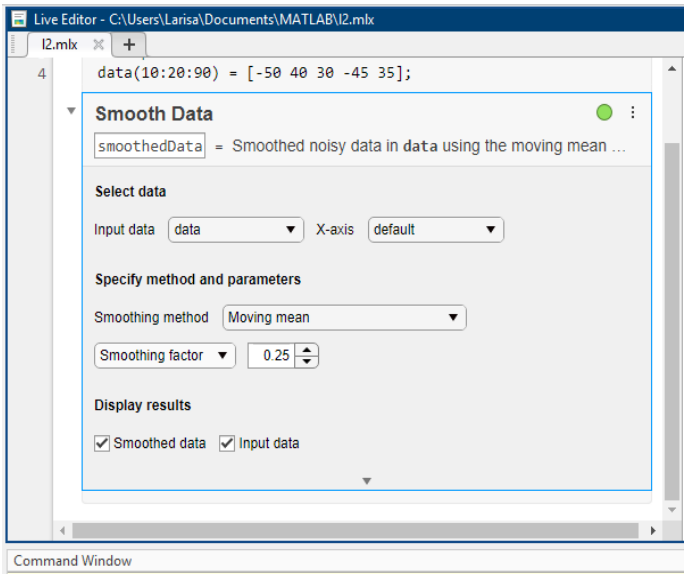


Рис. 8.11. Диалоговое окно инструмента *Smooth Data*

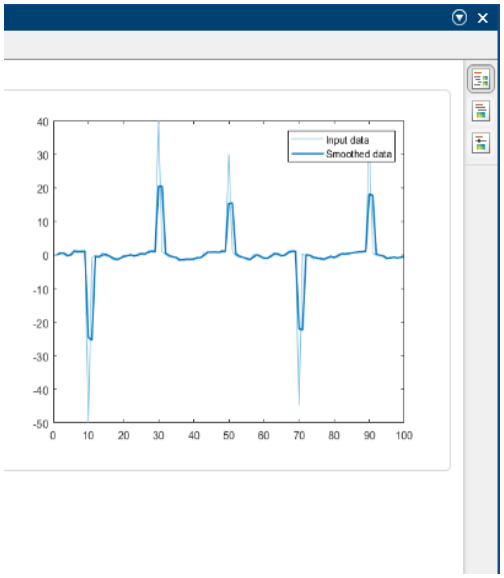


Рис. 8.12. Результат сглаживания данных инструментом *Smooth Data*

Продолжим анализ данных. Посмотрим, есть ли в данных резкие изменения в среднем значении, отклонении или наклоне и прерывании данных, т. е. найдем локальные экстремумы.

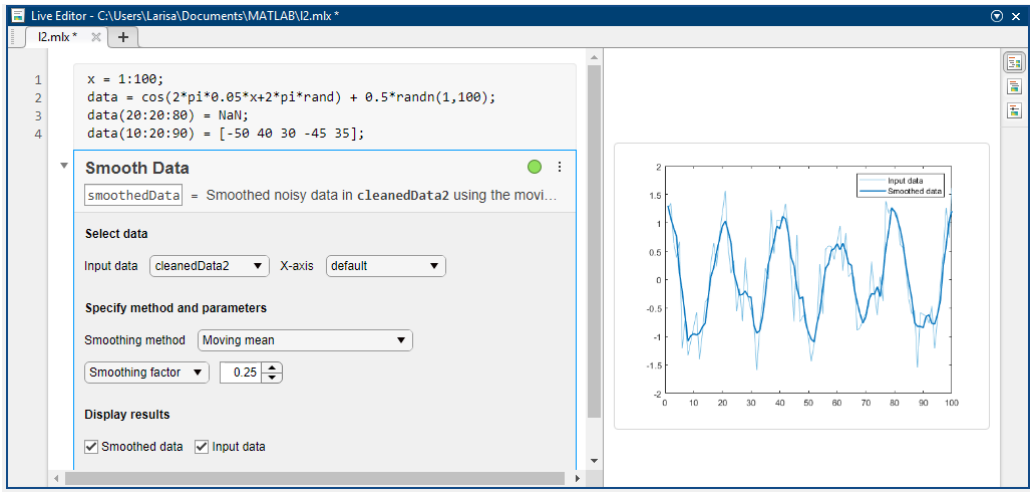


Рис. 8.13. Данные после сглаживания инструментом *Smooth Data*

Поиск осуществляется с помощью инструмента *Find Local Extrema* во вкладке *Task* (рис. 8.14). В качестве входных данных выберем результаты работы инструмента *Smooth Data* – smoothedData (рис. 8.15).

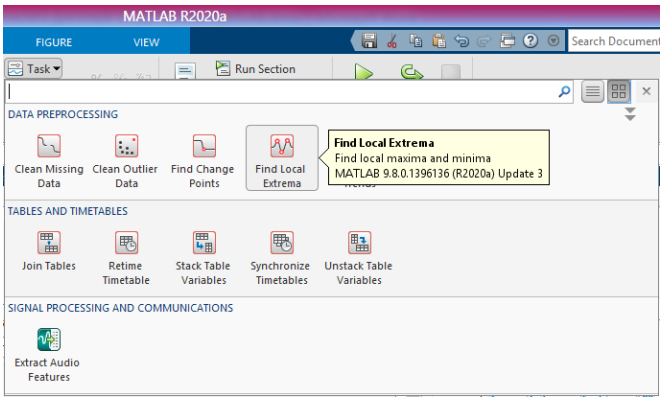


Рис. 8.14. Инструмент *Find Local Extrema*

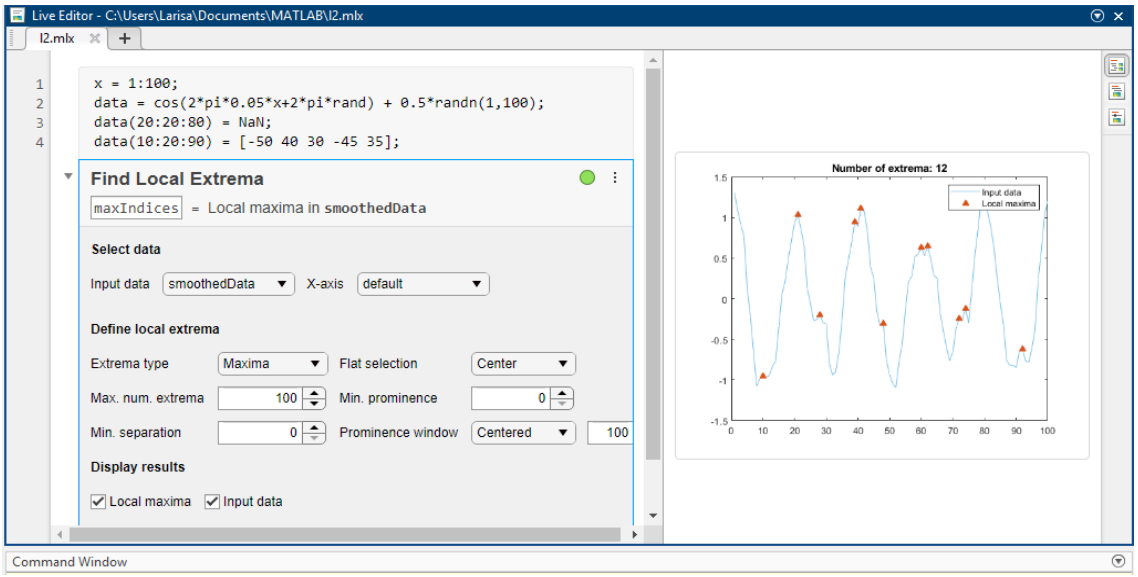


Рис. 8.15. Результат работы инструмента *Smooth Data*

Как видно по рис. 8.15, инструментом обнаружено 12 точек экстремума.

Таким образом, с помощью инструментов *Live Editor* удалось исключить из анализа значения NaN, сгладить выбросы, найти точки экстремума, т. е. подготовить данные для последующего анализа.

Рассмотренный способ предобработки данных не единственный. То же можно выполнить, написав соответствующий программный код. Рассмотрим примеры.

8.2. Поиск отсутствующих значений вручную

Создадим числовой массив, содержащий, наряду с числовыми данными, пропуски (NaN):

x = [NaN 1 2 3 4];

Один из способов найти значения NaN в данных – воспользоваться isnan- или ismissing-функциями, которые возвращают логический массив, указывающий на местоположение любого значения NaN.

Синтаксис следующий:

isnan (A)

Функция **isnan** проверяет, являются ли элементами массива значения NaN. Возвращает массив одного размера с массивом *A*, содержащий логическую единицу (1, TRUE) там, где значением элемента является NaN, и логический ноль (0, FALSE) там, где NaN не является значением элемента.

В нашем примере фрагмент кода live-скрипта будет выглядеть следующим образом:

x = [NaN 1 2 3 4];
TF = isnan(x)
TF = 1 * 5 logical array
1 0 0 0 0

В случае других типов данных, например строк:

x = ["a" "b" "c" "d" missing];
TFstring = ismissing(x)
TFstring = 1 * 5 logical array
0 0 0 0 1

Если импортируемые данные представляют собой переменные нескольких типов данных, можно найти все отсутствующие значения одним вызовом функции **ismissing** независимо от их типа, например командой

xTable = table (xDouble', xDatetime', xString', xCategorical')

8.3. Обработка недостающих данных вручную

Набор данных может содержать значения, которые пользователь хочет обработать как недостающие данные, но при этом они не являются стандартными отсутствующими значениями MATLAB, такими как NaN. В этом случае для преобразования этих значений в стандартные отсутствующие значения можно использовать функцию **standardizeMissing**.

Функция **standardizeMissing ()** вставляет стандартные отсутствующие значения в зависимости от типа данных:

– для double – NaN;

– для datetime – NaT;

– для string – <missing>;

– для categorical – <undefined>;

– для char – ' ';

– для cell из символьных массивов – {''}.

Синтаксис следующий:

```
B = standardizeMissing (A, indicator)
B = standardizeMissing (A, indicator, 'DataVariables', datavars)
```

где *A* – массив, indicator – стандартное значение для замены, datavars – имена переменных, в которых необходимо заменить отсутствующие значения.

Например, обработаем значение «5» как недостающее значение NaN:

```
x = [5 1 2 3 5];
xStandard = standardizeMissing(x,[5 NaN])
xStandard = 1 * 5
NaN 1 2 3 NaN
```

MATLAB также позволяет сохранить отсутствующие значения как часть импортированного набора данных, но выделить их из остальной части данных. Несколько функций MATLAB предоставляют возможность управлять размещением отсутствующих значений перед последующей обработкой, например воспользоваться опцией 'MissingPlacement' функции **sort**, перемещающей значение NaN в конец данных. Соответствующий фрагмент кода:

```
x = [5 1 2 3 5];
xStandard = standardizeMissing(x,[5 NaN])
xSort = sort(xStandard,'MissingPlacement','last')
%вид массива до применения опции MissingPlacement
xStandard = 1 * 5
NaN 1 2 3 NaN
%вид массива после применения опции MissingPlacement
xStandard = 1 * 5
1 2 3 NaN NaN
```

8.4. Удаление, замена и игнорирование отсутствующих значений вручную

Отсутствующие значения могут представлять собой данные, неприменимые для обработки или анализа. В этом случае их можно полностью удалить или заменить на другое значение.

Для удаления используется функция **rmmissing**. Синтаксис следующий:

```
R = rmmissing (A)
R = rmmissing (A, dim)
R = rmmissing (___, Name, Value)
[R,TF] = rmmissing (___)
```

Команда **R = rmmissing (A)** удаляет недостающие записи из массива или таблицы. Если *A* – вектор, функция **rmmissing** удаляет любую запись, которая содержит недостающие данные. Если *A* – матрица или таблица, функция **rmmissing** удаляет любую строку, которая содержит недостающие данные.

В команде **R = rmmissing (A, dim)** dim задает размерность, в которой будет «работать» функция. По умолчанию функция **rmmissing** действует по первому измерению, размер которого не равняется единице.

Команда **R = rmmissing (___, Name, Value)** задает дополнительные параметры для удаления недостающих записей с помощью одного или нескольких аргументов пары «имя – значение». Например, можно использовать функцию **rmmissing (A, 'MinNumMissing', n)** для удаления строки матрицы *A*, содержащей по крайней мере *n* отсутствующих значений.

Команда **[R,TF] = rmmissing (___)** также возвращает логический вектор, соответствующий строкам или столбцам массива *A*, в котором были удалены недостающие значения.

В примере ниже отсутствующие значения перемещаются в конец вектора, а затем удаляются функцией **rmmissing**:

```
x = [5 1 2 3 5];
xStandard = standardizeMissing(x, [5 NaN])
xSort = sort(xStandard,'MissingPlacement', 'last')
xRemove = rmmissing (xStandard)
xRemove = 1 * 3
1 2 3
```

Для заполнения отсутствующих значений используется функция **fillmissing**. Синтаксис следующий:

F = fillmissing (A, 'constant', v)
F = fillmissing (A, method)
F = fillmissing (A, movmethod, window)
F = fillmissing (A, fillfun, gapwindow)
F = fillmissing (___, dim)
F = fillmissing (___, Name, Value)
[F,TF] = fillmissing (___)

Команда **F = fillmissing (A,'constant', v)** заменяет недостающие записи массива или таблицы постоянным значением *v*. Если *A* – матричный или многомерный массив, *v* может быть скаляром или вектором. В случае, если *v* – вектор, каждый элемент *v* задает значение для замены в соответствующем столбце *A*. Если *A* – таблица или расписание, *v* может также быть массивом ячеек, элементы которого содержат значения для каждой табличной переменной.

Отсутствующие значения заменяются согласно следующим типам данных *A*:

– NaN – 'double';
– NaT – datetime;
– <missing> – строка;
– <undefined> – категориальный;
– ' ' – 'char';
– {''} ячейка из символьных массивов.

Команда **F = fillmissing (A, method)** заменяет недостающие данные с помощью метода, заданного параметром method. Например, функция fillmissing (A, 'previous') заменит недостающие записи предыдущей недостающей записью A.

Команда **F = fillmissing (A, movmethod, window)** заменяет недостающие записи с помощью скользящего среднего значения окна или медианы с длиной окна window. Например, функция fillmissing (A, 'movmean', 5) заменяет данные с помощью скользящего среднего значения с длиной окна 5.

Команда **F = fillmissing (A, fillfun, gapwindow)** заменяет недостающие записи с помощью пользовательского метода, заданного указателем на функцию **fillfun**, и фиксированного окна, окружающего каждый разрыв. Функция **fillfun** должна иметь входные параметры xs,

ts и tq – векторы, содержащие выборочные данные xs из gapwindow о длине, местоположении недостающих записей.

Команда **F = fillmissing (___, dim)** задает размерность массива *A*, по которой будут заполняться пропуски. По умолчанию функция **fillmissing** действует по первому измерению массива, размер которого не равняется единице. Например, если *A* – матрица, функция fillmissing (A, 2) заполнит недостающие строки данными столбца.

Команда **F = fillmissing (___, Name, Value)** задает дополнительные параметры для заполнения отсутствующих значений с помощью одного или нескольких аргументов пары «имя – значение». Например, если *t* – вектор из временных стоимостей, функция fillmissing (A, 'linear', 'SamplePoints', t) интерполирует данные в *A* относительно времен в *t*.

Команда **[F, TF] = fillmissing (___)** также возвращает логический массив, соответствующий записям *A*.

С помощью функции **fillmissing** заменим отсутствующие значения, например, на ноль:

```
x = [5 1 2 3 5];
xStandard = standardizeMissing(x,[5 NaN])
xFill = fillmissing(xStandard,'constant',0)
xFill = 1 * 5
0 1 2 3 0
```

Если нет необходимости явным образом определять местоположение, заполнять или удалять недостающие данные, MATLAB позволяет просто проигнорировать значения NaN.

Например, если вычисляется сумма вектора, содержащего значения NaN, результатом является NaN. При помощи опции 'omitnan' функции **sum** можно проигнорировать NaN непосредственно в сумме:

```
x = [nan nan 2 3 5];
sumOmitnan = sum(x,'omitnan')
sumOmitnan = 10
```

8.5. Удаление, замена и игнорирование выбросов вручную

Функция **rmoutliers** позволяет обнаружить и удалить выбросы из данных. Синтаксис следующий:

```
B = rmoutliers (A)
B = rmoutliers (A, method)
```

B = rmoutliers (A, 'percentiles', threshold)
B = rmoutliers (A, movmethod, window)
B = rmoutliers (___, dim)
B = rmoutliers (___, Name, Value)
[B,TF] = rmoutliers (___)

Команда **B = rmoutliers (A)** обнаруживает и удаляет выбросы из данных в векторе, матрице, таблице или расписании. По умолчанию выброс является значением большим, чем три масштабируемых средних абсолютных отклонения (MAD).

В примере ниже в векторе *A* значения 310 и 300 превышают в три раза MAD, поэтому будут опознаны системой MATLAB как выбросы. Следующий скрипт удалит эти значения:

```
A = [57 59 60 310 59 58 57 58 300 61 62 60 62 58 57];
disp ('вектор до удаления выбросов');
A
B = rmoutliers (A);
disp ('вектор после удаления выбросов');
B
```

```
>> rmout
вектор до удаления выбросов
A =
57   59   60  310   59   58   57   58  300   61   62   60   62   58   57
вектор после удаления выбросов
B =
57   59   60   59   58   57   58   61   62   60   62   58   57
```

Команда **B = rmoutliers (___, dim)** удаляет выбросы по измерению вектора (или матрицы, таблицы, расписания). Например, следующий код удалит выбросы (310 и 300) по столбцам вектора *A*:

```
A = [57 59 60 310 59 58 57 58 300 61 62 60 62 58 57];
disp ('вектор до удаления выбросов');
A
C = rmoutliers(A,2);
disp ('вектор после удаления выбросов');
C
```

```
вектор до удаления выбросов
A =
57   59   60  310   59   58   57   58  300   61   62   60   62   58   57
```

вектор после удаления выбросов
C =
57 59 60 59 58 57 58 61 62 60 62 58 57

Приведем еще пример. Создадим матрицу *A* с размерами 5 × 5. Зададим значение A(5,5) равным выбросу. Следующий код удалит строку матрицы, содержащую выброс:

```
disp ('матрица до удаления выбросов');
A = magic(5);
A(5,5)=400;
A
disp ('матрица после удаления выбросов');
C = rmoutliers(A,1)
>> rmout
```

матрица до удаления выбросов
A =

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	400

матрица после удаления строки, содержащей выброс
C =

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3

Команда **B = rmoutliers (A, method)** задает метод для определения выбросов. Например, функция **rmoutliers (A, 'mean')** определяет выбросы как элементы вектора *A*, большие трех стандартных отклонений от среднего значения.

Если *A* – строка или вектор-столбец, функция **rmoutliers** обнаруживает выбросы и удаляет их. Если *A* – матрица, таблица или расписание, функция **rmoutliers** обнаруживает выбросы в каждом столбце *A* отдельно и удаляет целую строку.

Например, создадим вектор *A*, содержащий два выброса. Удалим их. Вектор *TF* позволяет идентифицировать, какие элементы

входного вектора были обнаружены как выбросы и удалены. Преобразованный вектор *A* сохраним в векторе *B*:

```
A = [57 59 60 100 59 58 57 58 300 61 62 60 62 58 57];
[B,TF] = rmoutliers(A);
B = 1×13
57   59   60   59   58   57   58   61   62   60   62   58   57
TF = 1x15 logical array
0   0   0   1   0   0   0   0   1   0   0   0   0   0   0
A(TF)
ans = 1×2
100   300
```

Визуализировать векторы *A*, *B*, *TF* можно иначе: выбрав соответствующее имя в рабочей области (рис. 8.16, 8.17).

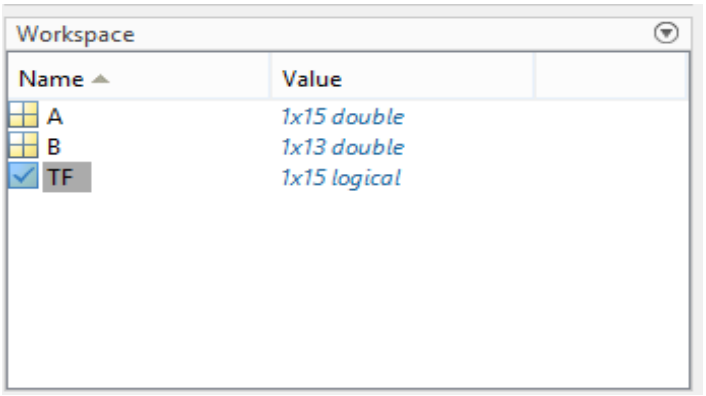


Рис. 8.16. Визуализация переменных рабочей области

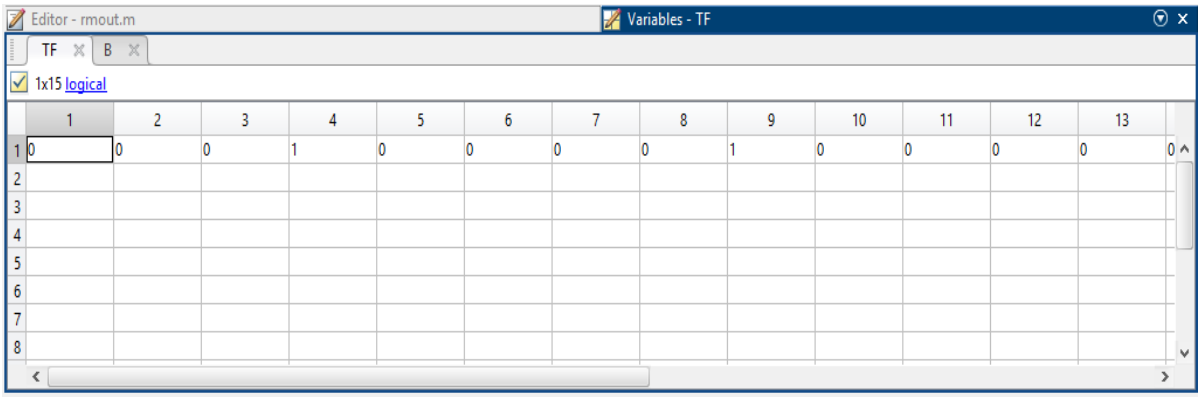


Рис. 8.17. Визуализация переменных типа «таблица» рабочей области

Вопросы для самоконтроля

1. Кратко охарактеризуйте каждый способ заполнения пропусков инструментом *Clean Missing Data*. Обоснуйте, в каких случаях какой способ лучше использовать.

2. Кратко охарактеризуйте способы поиска, сглаживания и удаления выбросов инструментом *Clean Outlier Data*. Обоснуйте, в каких случаях какой способ лучше использовать.

3. Кратко охарактеризуйте инструменты *Find Change Points* и *Find Local Extrema*. Чем они различаются?

Практическая работа

ПРЕДОБРАБОТКА ИМПОРТИРОВАННЫХ ДАННЫХ СРЕДСТВАМИ ИНСТРУМЕНТА LIVE EDITOR

1. С помощью специальной функции проверьте данные таблицы T1 из практической работы «Импорт данных» (гл. 7) на наличие пропусков.

2. Заполните пропуски с помощью одного из инструментов *Live Editor*. Обоснуйте, почему использовали именно этот инструмент.

3. Сохраните и визуализируйте данные после обработки пропусков.

4. С помощью специальной функции проверьте данные на наличие выбросов.

5. По ситуации не изменяйте, сгладьте или удалите выбросы. Обоснуйте свое решение.

6. Сохраните и визуализируйте данные после обработки выбросов.

7. Найдите и визуализируйте точки экстремума. Что они характеризуют?

8. Сохраните обработанные данные в таблице T2.

9. Оформите отчет и представьте преподавателю.

Глава 9
АНАЛИЗ ДАННЫХ

Анализ данных начнем с описания операций системы MATLAB, которые предназначены для анализа и обработки данных, заданных в виде числовых массивов.

9.1. Основные операции анализа и обработки числовых массивов данных

К основным операциям относят:

– sum, cumsum – суммирование элементов массива;

– prod, cumprod – произведение элементов массива;

– sort – сортировка элементов массива по возрастанию;

– max – определение максимальных элементов массива;

– min – определение минимальных элементов массива;

– median – определение срединных значений (медиан) элементов массива;

– mean – определение средних значений элементов массива;

– cov – определение ковариационной матрицы элементов массива;

– std – определение стандартных отклонений элементов массива.

Рассмотрим некоторые из них.

Функции **sum, cumsum** – суммирование элементов массива. Синтаксис следующий:

```
sx = sum (X)
csx = cumsum (X)
```

Команда **sx = sum (X)** в случае одномерного массива возвращает сумму элементов массива, в случае двумерного массива возвращает вектор-строку, содержащий суммы элементов каждого столбца.

Команда **csx = cumsum (X)**, кроме того, возвращает все промежуточные результаты суммирования.

Рассмотрим массив *M* с размерами 3 × 3:

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

```
M=[8 1 6; 3 5 7; 4 9 2];
sx = sum (M)
M=[8 1 6; 3 5 7; 4 9 2];
```

```
csx = cumsum (M)
sx =
   15    15    15
%вывод последовательных сумм в столбцах
csx =
    8     1     6
   11     6    13
   15    15    15
```

Функции **prod, cumprod** – произведение элементов массива. Синтаксис следующий:

```
px = prod (X)
cpx = cumprod (X)
```

Команда **px = prod (X)** в случае одномерного массива возвращает произведение элементов массива, в случае двумерного массива возвращает вектор-строку, содержащий произведения элементов каждого столбца.

Команда **cpx = cumprod (X)**, кроме того, возвращает все промежуточные результаты.

Рассмотрим массив M = magic(3),

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

```
M=[8 1 6; 3 5 7; 4 9 2];
px = prod(M)
cpx = cumprod(M)
px =
   96    45    84
cpx =
    8     1     6
   24     5    42
   96    45    84
```

Функция **sort** – сортировка элементов массива по возрастанию. Синтаксис следующий:

```
Y = sort (X)
[Y, I] = sort (X)
```

Команда **Y = sort (X)** в случае одномерного массива упорядочивает элементы массива по возрастанию, в случае двумерного массива происходит упорядочение элементов каждого столбца. Например:

```
M=[8 1 6; 3 5 7; 4 9 2];
y=sort (M)
y =
   3   1   2
   4   5   6
   8   9   7
```

Команда **[Y, I] = sort (X)**, кроме массива упорядоченных элементов по столбцам, возвращает массив индексов. Например:

```
M=[8 1 6; 3 5 7; 4 9 2];
y=sort (M)
[Y, I] = sort (M)
I =
   2   1   3
   3   2   1
   1   3   2
```

Возвращение индексов позволяет восстановить структуру исходного массива. Если анализируемый массив содержит комплексные элементы, то сортировка выполняется для массива abs (X).

Функция **max** – определение максимальных элементов массива. Синтаксис следующий:

```
Y = max (X)
[Y, I] = max (X)
C = max (A, B)
```

Команда **Y= max (X)** в случае одномерного массива возвращает наибольший элемент; в случае двумерного массива возвращает вектор-строку, содержащий максимальные элементы каждого столбца. Таким образом, max (max (X)) – это наибольший элемент массива.

Команда **[Y, I] = max (X)**, кроме самих максимальных элементов, возвращает вектор-строку индексов этих элементов в данном столбце.

Команда **C = max (A, B)** возвращает массив *C* с теми же размерами, какие имеют массивы *A* и *B*, каждый элемент массива *C* есть максимальный из соответствующих элементов массивов *A* и *B*.

Если анализируемый массив содержит комплексные элементы, то максимальные элементы определяются из условия max (abs (X)). Если массив содержит один или несколько элементов типа NaN, то результатом операции будет NaN.

Например, для массива M = [8 1 6; 3 5 7; 4 9 2] результат выполнения функций будет следующим:

y = max (M)	[y, I] = max (M)	max (max (M))
y = 8 9 7	y = 8 9 7 I = 1 3 2	9

Функция **min** – определение минимальных элементов массива. Синтаксис следующий:

Y = min (X)

[Y, I] = min (X)

C = min (A, B)

Команда **Y = min (X)** в случае одномерного массива возвращает наименьший элемент, в случае двумерного массива возвращает вектор-строку, содержащий минимальные элементы каждого столбца. Таким образом, min (min (X)) – это наименьший элемент массива.

Команда **[Y, I] = min (X)**, кроме самих минимальных элементов, возвращает вектор-строку индексов этих элементов в данном столбце.

Команда **C = min (A, B)** возвращает массив *C* с теми же размерами, какие имеют массивы *A* и *B*, каждый элемент массива *C* есть минимальный из соответствующих элементов массивов *A* и *B*.

Если анализируемый массив содержит комплексные элементы, то минимальные элементы определяются из условия min (abs (X)). Если массив содержит один или несколько элементов типа NaN, то результатом операции будет NaN.

Например, для массива M = [8 1 6; 3 5 7; 4 9 2] результат выполнения функций будет следующим:

y = min (M)	[y, I] = min (M)	min (min (M))
y = 3 1 2	y = 3 1 2 I = 2 1 3	1

Функция **median** – определение срединных значений (медиан) элементов массива. Синтаксис следующий:

mdx = median (X)

Команда **mdx = median (X)** в случае одномерного массива возвращает значение срединного элемента, в случае двумерного массива возвращает вектор-строку, содержащий значение срединных элементов каждого столбца. Таким образом, median (median (X)) – это сре-

динный элемент (медиана) массива, что совпадает со значением median (X(:)). Для массива M = magic(4) результат будет следующим:

```
M   16   2    3    13
     5   11   10   8
     9   7    6    12
     4   14   15   1
```

Для команды **mdx = median (M)** результат будет 7 9 8 10.

Для команды **median (median (M))** результат будет 8.5000.

Функция **mean** – определение средних значений элементов массива. Синтаксис следующий:

mx = mean (X)

Команда **mx = mean(X)** в случае одномерного массива возвращает арифметическое среднее элементов массива, в случае двумерного массива возвращает вектор-строку, содержащий арифметическое среднее элементов каждого столбца. Таким образом, mean (mean (X)) – это арифметическое среднее (математическое ожидание) элементов массива, что совпадает со значением mean (X(:)). Например, рассмотрим массив M = magic(4):

```
M   16 2   3   13
     5  11  10  8
     9  7   6   12
     4  14  15  1
```

```
mx = mean (M)                          mean (mean(M))
mx = 8.5000 8.5000 8.5000 8.5000       ans = 8.5000
```

Функция **std** – определение стандартных отклонений элементов массива. Синтаксис следующий:

sx = std (X)

Команда **sx = std (X)** в случае одномерного массива возвращает стандартное отклонение элементов массива, в случае двумерного массива возвращает вектор-строку, содержащий стандартное отклонение элементов каждого столбца. Например, рассмотрим массив M = magic(4):

```
M   16   2    3    13
     5   11   10   8
     9   7    6    12
     4   14   15   1
```

sx = std (M)
sx = 5.4467 5.1962 5.1962 5.4467

Функция **cov** – определение ковариационной матрицы элементов массива. Синтаксис следующий:
C = cov (X)
C = cov (X, y)

Команда **C = cov (X)** в случае одномерного массива возвращает дисперсию элементов массива, в случае двумерного массива, когда каждый столбец рассматривается как переменная, а каждая строка – как наблюдение, cov (X) возвращает матрицу ковариаций, diag (cov (X)) – вектор дисперсий, sqrt (diag (cov (X))) – вектор стандартных отклонений для каждого столбца.

Команда **C = cov (X, Y)**, где массивы *X* и *Y* имеют одинаковое количество строк, равносильна функции **cov ([X Y])**.

Например, рассмотрим массив M = magic(4)/norm(magic(4)):

M
0.4706	0.0588	0.0882	0.3824
0.1471	0.3235	0.2941	0.2353
0.2647	0.2059	0.1765	0.3529
0.1176	0.4118	0.4412	0.0294

C = cov (X)
				diag (cov (X))	sqrt (diag (cov (X)))
0.0257	-0.0239	-0.0222	0.0205	0.0257	0.01602
-0.0239	0.0234	0.0228	-0.0222	0.0234	0.1528
-0.0222	0.0228	0.0234	-0.0239	0.0234	0.01528
0.0205	-0.0222	-0.0239	0.0257	0.0257	0.1602

Вычисление матрицы ковариаций реализуется следующим алгоритмом:
```
[n, p] = size(X);
X = X - ones(n, 1) * mean(X);
C = X' * X/(n - 1);
```

9.2. Выбор трендовой модели, оценка модели, симуляция и предсказание

На первом этапе в результате предварительной обработки данные приводятся к единообразному виду. Далее необходимо привести их к так называемому стационарному виду, т. е. убедиться, что в них отсутствует сезонная составляющая.

Определение наличия компоненты сезонности необходимо для того, чтобы входная информация обладала свойством репрезентативности (соответствие характеристик выборки характеристикам популяции или генеральной совокупности в целом).

Ряд можно считать сезонным, если при рассмотрении его внешнего вида можно сделать предположение о повторяемости формы кривой через равные промежутки времени.

Рассмотрим два графика, отражающих объем продаж по двум группам товаров – краски и автозапчастей (рис. 9.1, 9.2). По каждому графику сделаем выводы, почему существует именно такое поведение продаж по определённой группе товаров.

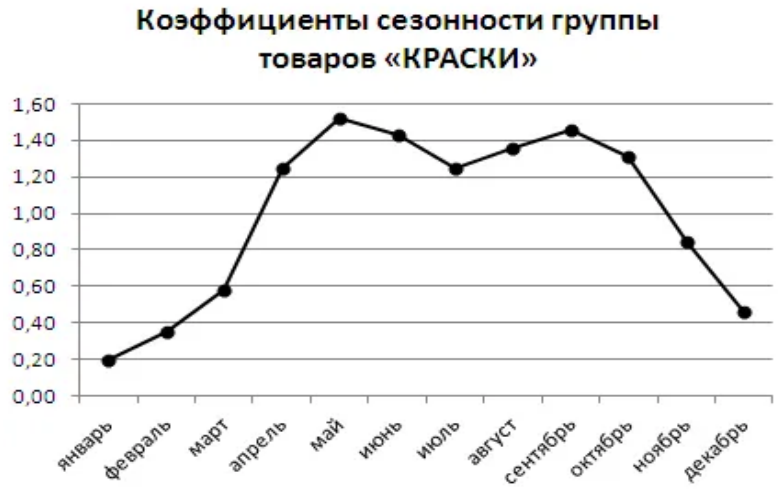


Рис. 9.1. Объем продаж красок

По графику на рис. 9.1 можно утверждать, что данная группа товаров имеет ярко выраженную сезонность, о чем свидетельствуют резкие всплески, т. е. между минимальным коэффициентом сезонности и максимальным коэффициентом существует большая разница. Продажи компании в январе находятся на самом минимальном уровне (коэффициент сезонности 0,2), так как зимой люди отмечают новогодние праздники. Это не самое подходящее время для ремонта и покраски поверхностей (заборов, гаражей, стен в квартире и т. д.).

В феврале и марте наблюдается постепенный рост коэффициентов сезонности, который говорит о том, что продажи немного выросли. Обратите внимание: темп роста продаж небольшой. И это также объясняется в первую очередь низкой температурой за окном, которая сдерживает людей от проведения ремонтов и строительства домов.

С наступлением апреля происходит бурный рост продаж красок. Теперь стройки активизировались, ремонты делаются всё чаще, так как наступила благоприятная погода для проведения такого рода работ.

В мае наблюдается максимальный пик в продажах.

С наступлением лета продажи незначительно падают, что объясняется массовыми поездками людей на отдых. Итак, в июне и июле продажи падают, но незначительно, так как всё равно за окном благоприятная погода не только для отпусков, но и для ремонтов в квартирах.

В августе население всё реже уезжает на отдых, что сразу влияет на рост продаж. В сентябре и октябре продажи держатся приблизительно на уровне, который был зафиксирован ещё в апреле, когда наступила тёплая погода. С наступлением холодов в ноябре продажи падают и в декабре практически достигают уровня продаж, который был зафиксирован в начале года.

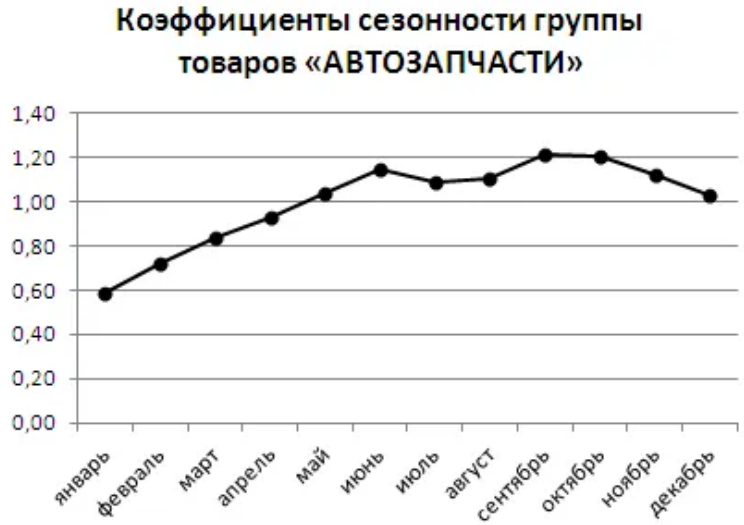


Рис. 9.2. Объем продаж автозапчастей

По графику на рис. 9.2 можно сказать, что группа товаров «Автозапчасти» не имеет ярко выраженной сезонности, которая наблюдается с группой товаров «Краски». Здесь в течение года происходит незначительный рост коэффициентов сезонности от 0,6 (в январе) до 1,2 (в сентябре).

Если давать объяснение продажам автозапчастей, то становится очевидным, что с наступлением морозов продажи падают. Так, после новогодних праздников в январе продажи начинают свой рост с минимальной отметки. Постепенно с улучшением погоды увеличивается

и темп продаж автозапчастей. С весны многие автолюбители приступают к плановому ремонту своего автомобиля после зимнего периода, и рост продаж продолжается до июня включительно. Затем, как и в случае с красками, продажи незначительно падают, что объясняется сезоном отпусков. После отпусков продажи автозапчастей снова увеличиваются и находятся на пике в сентябре и октябре, затем продажи падают из-за понижения температуры за окном.

Как видно из приведенных примеров, самый простой способ определения сезонности – визуализация выборки данных.

Если по графику сложно выявить сезонность в ряду, сделать это можно аналитически с помощью так называемых индексов сезонности, которые могут рассчитываться по-разному.

По данным одного года индексы сезонности рассчитываются как отношение помесячных (или квартальных) уровней к среднему уровню за год, т. е. $J_{\text{сез}} = \frac{y_i}{\bar{y}}$, где y_i – помесячный (или квартальный) уровень; $\bar{y}$ – средний уровень за год.

Для большей надежности сезонность изучается по данным за три года. В этом случае для каждого месяца рассчитывается средний уровень за три года, который и сопоставляется со средним уровнем за весь период, т. е. $J_{\text{сез}} = \frac{\overline{y_i}}{\bar{y}}$, где $\overline{y_i}$ – средний уровень для каждого месяца; $\bar{y}$ – средний уровень за весь период.

Рассмотрим применение описанных способов на линейном и параболическом типах трендовых моделей.

Линейная трендовая модель (линейный тренд). Применяется для прогнозирования временных рядов, данные в которых увеличиваются (или убывают) с постоянной скоростью. Формально описывается функцией $y = k_1x + b$, где y – последовательность анализируемых значений (например, продажи акций по месяцам); k_1 – угловой коэффициент временного ряда; x – номер периода во временном ряду (например, номер месяца, квартала, дня); b – свободный член (минимальный уровень анализируемого показателя). Если $k_1 > 0$, динамика тренда положительная (например, продажи акций растут). Если $k_1 < 0$, то динамика тренда отрицательная.

Рассмотрим применение линейного тренда для построения в MATLAB прогноза продаж автомобилей на вторую половину года на основе данных о продажах за первую половину года. Для простоты

зададим значения данных, в которых отсутствует сезонная составляющая.

Примем за *y* объем продаж автомобилей (анализируемый показатель), за *x* – номера месяцев в анализируемом периоде. Создадим timeseries-объект.

I этап. Визуализируем данные с помощью графика (рис. 9.3):

```
ts = timeseries([14560 15000  15810 16000 16200 16340], [1 2 3 4 5 6]);
plot (ts,'k')
title ('Объемы продаж');
xlabel ('месяцы');
ylabel('продажи');
```

Или таким способом (рис. 9.4):

```
y = [14560 15000  15810 16000 16200 16340];
ts = timeseries(y,1:6);
ts.Name = 'sales';
ts.TimeInfo.Units = 'months';
% начальная дата
ts.TimeInfo.StartDate = '01-Jan-2020';
% установка формата вывода даты
ts.TimeInfo.Format = 'mmm , yy';
% установить временную шкалу от даты начала
ts.Time = ts.Time - ts.Time (1)
plot(ts)
```

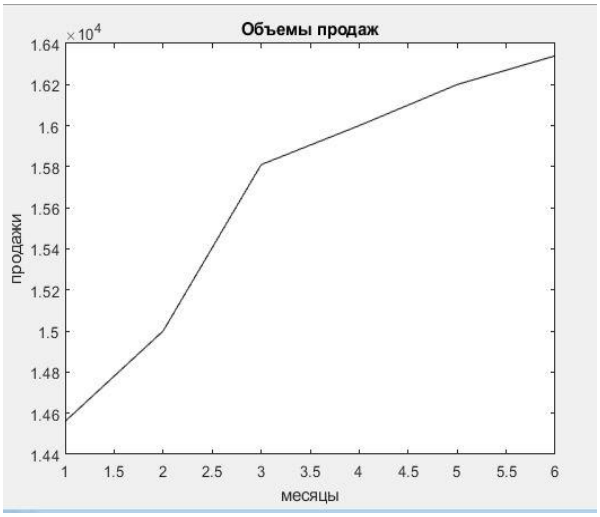


Рис. 9.3. Визуализация объема продаж автомобилей (1-й способ)

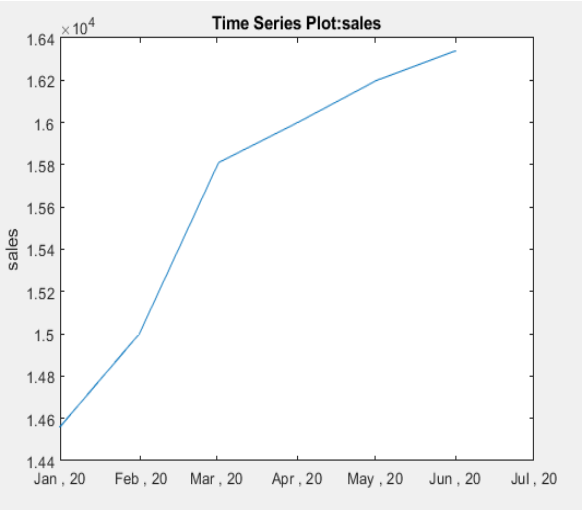


Рис. 9.4. Визуализация объема продаж автомобилей (2-й способ)

II этап. Выбор трендовой модели.

По графику (рис. 9.3, 9.4) можно сделать следующий вывод: хотя временной ряд и колеблется, визуально значения ряда имеют тенденцию увеличиваться во времени, форма кривой периодически не повторяется, т. е. ряд не сезонный, с ясным восходящим трендом. Это признак линейной трендовой модели.

III этап. Оценка модели.

Подтвердим предположение аналитически. Рассчитаем индекс сезонности для каждого месяца. Для этого с помощью функции **mean ()** вычислим индекс продаж: отношение продаж месяца к средней величине продаж (INDEX).

```
y = [14560 15000 15810 16000 16200 16340];
ts = timeseries(y,1:6);
M=mean(ts)
M =
  1.5652e+04
```

Таким образом, среднее значение объема продаж за шесть месяцев составляет 15 652 автомобиля.

Индекс продаж (INDEX) и среднее значение продаж (M_I):

```
y = [14560 15000 15810 16000 16200 16340];
ts = timeseries(y,1:6);
M=mean(ts);
INDEX=y/M
M_I=mean(INDEX)
INDEX =
   0.9303    0.9584    1.0101    1.0223    1.0350    1.0440
M_I =
    1
```

Значение INDEX подтверждает увеличение значения данных с постоянной скоростью, что свидетельствует о несезонном тренде. Следовательно, вывод, сделанный по визуальной модели, правильный и тренд не сезонный линейный.

Построим линию тренда (рис. 9.5):

```
ts = timeseries([14560 15000  15810 16000 16200 16340], [1 2 3 4 5 6]);
plot (ts,'.')
lsline    % линия тренда
title ('Объемы продаж');
xlabel ('месяцы');
ylabel('продажи');
```

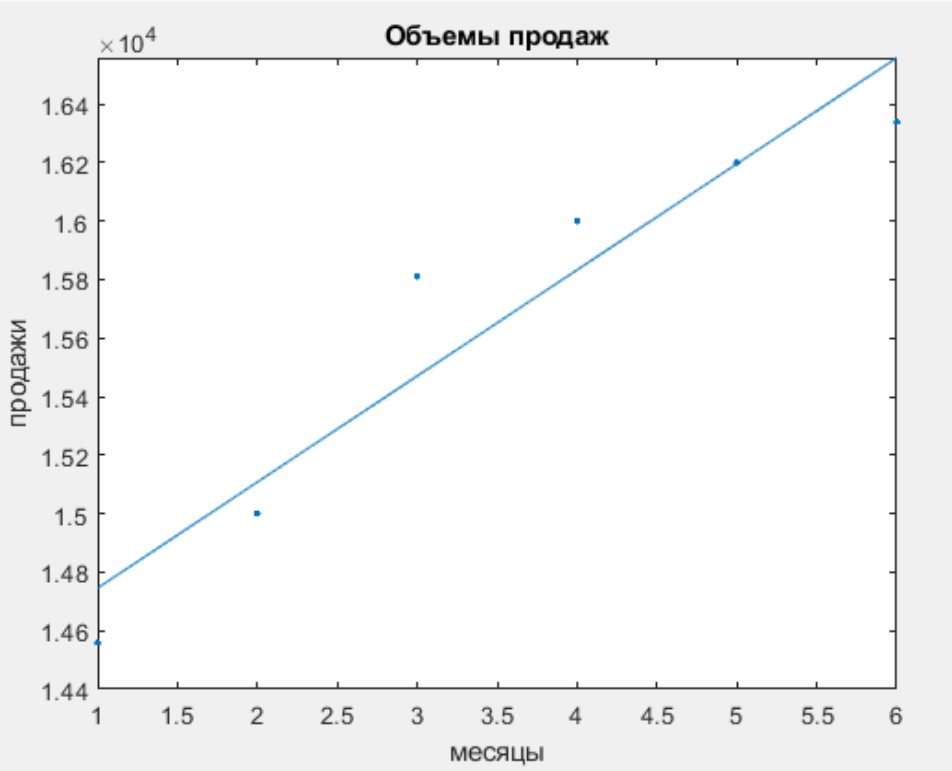


Рис. 9.5. Визуализация тренда

Выделим уравнение тренда из данных. Сделать это можно разными способами. Один из способов – через функции **polyval ()** и **polyfit ()**.

Команда **p = polyfit (x, y, n)** находит коэффициенты полинома $p(x)$ степени n, который аппроксимирует функцию $y(x)$ методом наименьших квадратов. Выходом является строка p длиной $n + 1$, содержащая коэффициенты аппроксимирующего полинома. Функция **polyfit** «не умеет работать» с временными рядами, поэтому в качестве аргументов укажем векторы x и y.

По графику идентифицировали тренд как линейный, следовательно, нужен полином первой степени, значит, $n = 1$:

```
x=[1 2 3 4 5 6];
y = [14560 15000 15810 16000 16200 16340];
p=polyfit (x,y,1)
p =
   1.0e+04 *
    0.0363    1.4383
```

На выходе получили два числа – коэффициенты полинома, которые расположены в убывающем порядке. Соответствующее уравнение тренда в общем виде выглядит так: $y = p(2) + p(1)x$.

Получаем уравнение тренда: $y = 363x + 14\,383$.

Значит, объем продаж в выбранном промежутке времени растёт в среднем на 14 383 автомобиля в месяц.

Вычислим значения полинома в точках сетки:

```
f = polyval (p, x);
```

Построим график аппроксимирующего полинома на отрезке [1; 6] (рис. 9.6).

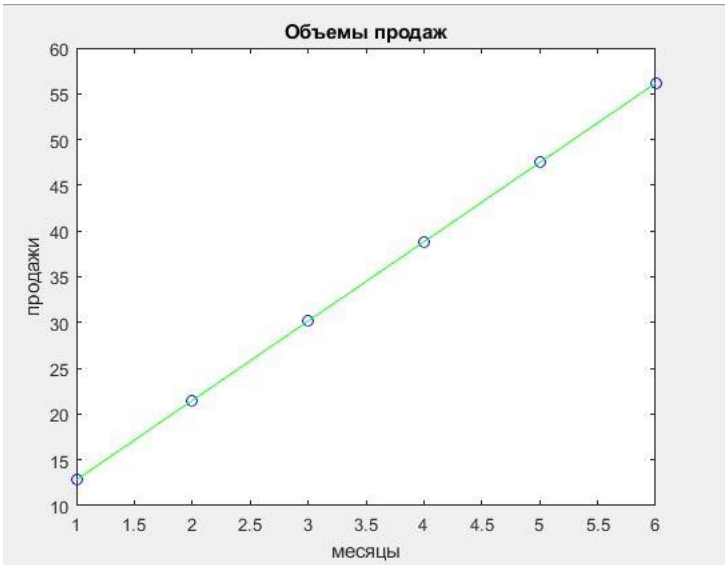


Рис. 9.6. График аппроксимирующего полинома

IV этап. Симуляция и предсказание.

Спрогнозируем значения тренда с седьмого по двенадцатый месяцы года по полученному уравнению с помощью функции **polyval ()**.

```
x=[1 2 3 4 5 6 7 8 9 10 11 12];
y= 363*x +14383;
p=polyfit (x, y, 1);
f = polyval (p, x);
plot (x, y, 'ob', x, f, '-g')
title ('Объемы продаж');
xlabel ('месяцы');
ylabel ('продажи');
```

Визуализируем данные (рис. 9.7).

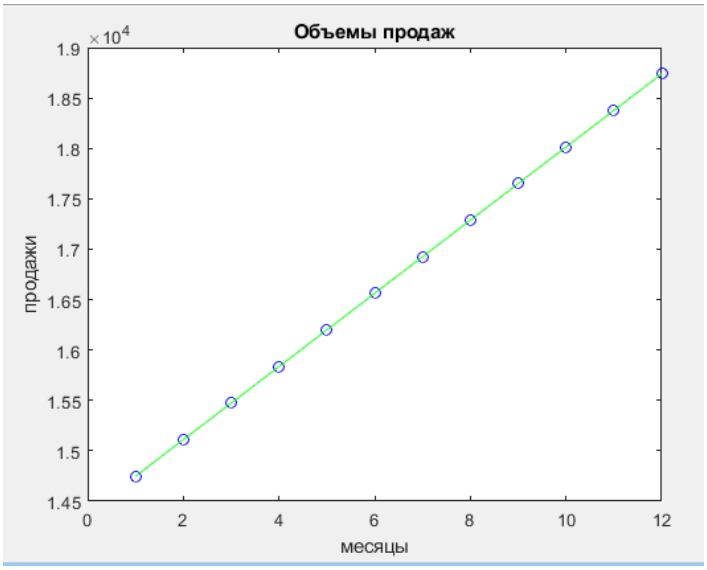


Рис. 9.7. Визуализация спрогнозированного тренда

По графику видно, что прогнозируемый объем продаж к концу года составит 18 739 автомобилей.

Параболическая трендовая модель (параболический тренд). В параболическом тренде, помимо уже упомянутых коэффициентов линейного тренда, появляется коэффициент, отвечающий за ускорение процесса, – k_2. Параболическая трендовая модель имеет вид $y = k_2x^2 + k_1x + b$. График этой функции – парабола с осью симметрии, параллельной оси ординат (рис. 9.8).

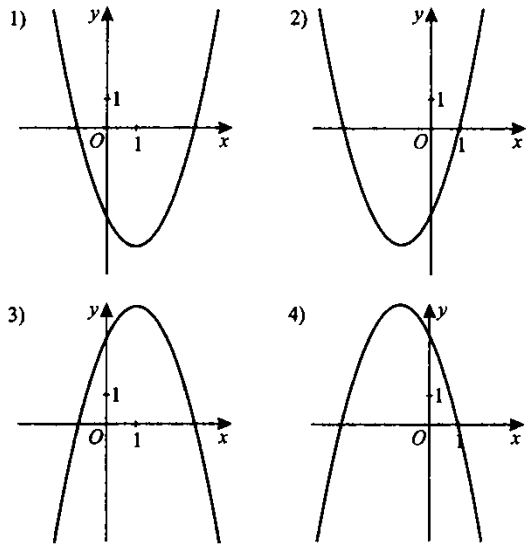


Рис. 9.8. Возможные расположения параболы
относительно осей координат

В указанной функции b – средний уровень тренда на момент (или период) времени, принятый за начало отсчета; k_1 – средний прирост за весь рассматриваемый период времени, не является константой, изменяется равномерно (см. линейную трендовую модель), со средним ускорением k_2 – константой и основным параметром параболы 2-го порядка.

Так как коэффициент b, как правило, величина положительная, то характер тренда определяется знаками параметров k_1 и k_2.

Характер функции обусловлен ее коэффициентами:

1) рост с ускорением: $k_2 > 0, k_1 > 0$ (рис. 9.9, *а*);

2) снижение с замедлением: $k_2 > 0$, $k_1 < 0$ (рис. 9.9, *б*);

3) рост с замедлением: $k_2 < 0$, $k_1 > 0$ (рис. 9.9, *в*);

4) снижение с ускорением: $k_2 < 0, k_1 < 0$ (рис. 9.9, *г*).

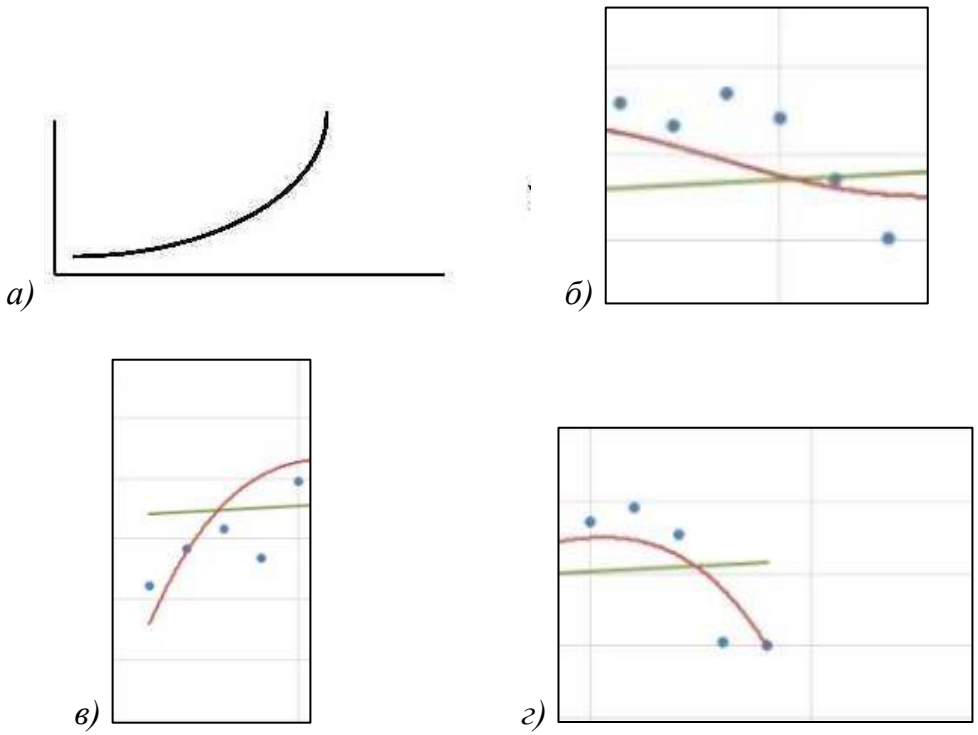


Рис. 9.9. Кривые трендовых уравнений

В качестве примера рассмотрим динамику товарооборота (тыс. руб.) между двумя фирмами в период с 2001 по 2012 г. Для простоты зададим значения данных, исключающие сезонность:

2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012
9494	9506	12 957	17 701	20 339	21 529	29 505	39 812	25 012	5	6	7

Зададим и визуализируем исходные данные в MATLAB (рис. 9.10):

```
y = [9494 9506 12957 17701 20339 21529 29505 39812 25012 5 6 7];
ts = timeseries(y,2001:2012);
plot (ts,'-')
title ('Динамика товарооборота');
xlabel ('года');
ylabel ('объем товарооборота');
```

Рис. 9.10. Графическое представление динамики товарооборота фирм

Вывод: визуально временной ряд имеет явно выраженные вершину и две ветви, что является признаками параболического тренда.

Рассчитаем коэффициенты k_2, k_1, b параболы:

```
y = [9494 9506 12957 17701 20339 21529 29505 39812 25012 5 6 7];
x=[2001 2001 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012];
format long;
p=polyfit (x,y,2)
p =
   1.0e+09 *
  -0.000000800534166   0.003211674927841  -3.221215745498336
```

Коэффициенты расположены в убывающем порядке, следовательно:

k2 = -0.000000800534166 *109, k1=0.003211674927841 *109,b= -3.221215745498336 *109.

Получаем уравнение параболы:

```
y= -800.534166 * power(x,2) + 3211674.927841 * x -
3221215745.498336;
```

Спрогнозируем значения тренда с 2013 по 2023 г. по полученному уравнению с помощью функции **polyval ()**. Визуализируем данные (рис. 9.11).

```
x=[2013 2014 2015 2016 2017 2020 2021 2022 2023];
y=-800.534166*power(x,2)+3211674.927841*x -3221215745.498336;
format long;
```

```
p=polyfit (x,y,2);
f = polyval(p, x);
plot(x, y, 'ob', x, f, '-g')
title ('Объемы товарооборота');
xlabel('года');
ylabel('товарооборот (тыс. руб.)');
```

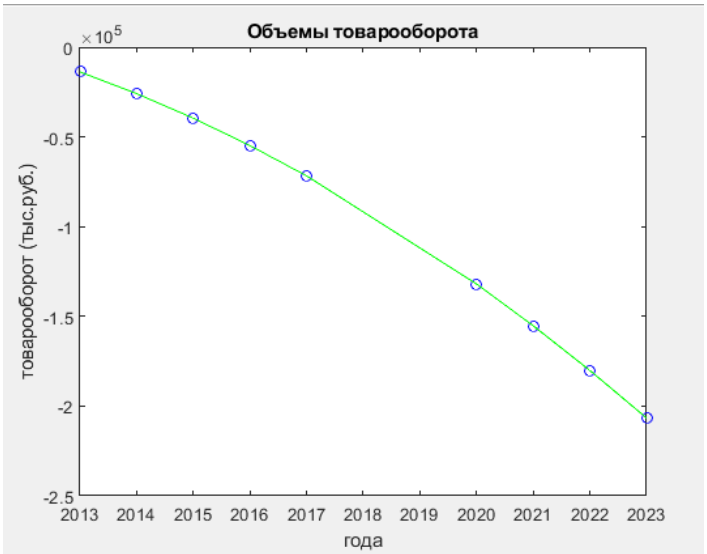


Рис. 9.11. Визуализация трендовой модели

Согласно построенной трендовой модели прогнозируемая динамика объемов товарооборота между фирмами в период с 2013 по 2023 г. отрицательная, характер – снижение с замедлением.

9.3. Методы выявления (сглаживания) основной тенденции в рядах данных

Основные составляющие временного ряда – тренд и сезонная компонента. А также значения временного ряда могут носить случайный характер.

Тренд – это некая тенденция, которая показывает изменение во времени какого-либо показателя(ей) под действием ряда факторов. Примерами тренда могут быть изменения цен на товары определенного направления, объекты строительства, изменения в моде или демографической ситуации и т. п.

Под действием других факторов изменения этого же показателя(ей) могут носить случайный характер, т. е. во временном ряду будет отсутствовать тенденция.

Тренд во временном ряду может иметь различную форму, с учетом которой выбирается соответствующий тип трендовой модели. Основные типы трендовых моделей (трендов) следующие: линейная, параболическая, степенная, показательная. В пособии ограничимся рассмотрением первых двух типов трендов как наиболее часто используемых на практике.

Сезонная составляющая ряда данных – это периодически повторяющаяся компонента ряда. Свойство сезонности означает, что примерно через равные промежутки времени форма кривой, которая описывает поведение зависимой переменной, повторяет свои характерные очертания. Свойство сезонности важно при определении количества ретроспективных данных, которые будут использоваться для прогнозирования.

Судить о наличии тенденции в ряду данных на основе его визуального анализа можно лишь тогда, когда четко видно, что при переходе от одного момента времени к другому уровни ряда возрастают или убывают. Однако, как правило, нельзя сразу сказать, есть тенденция или нет в изменении уровней ряда данных. Для определения тенденции применяются специальные методы.

К методам сглаживания ряда данных (T_t) с целью выявления основной тенденции развития относятся:

– метод укрупнения интервалов;

– метод наименьших квадратов;

– метод скользящей средней;

– аналитическое выравнивание динамических рядов.

Рассмотрим их подробнее.

9.3.1. Метод укрупнения интервалов

Применение метода укрупнения интервалов рассмотрим на основе данных тестирования скорости, Мбит/с, Интернета по месяцам, полученных сервисом сайта https://2ip.ru/speed и представленных ниже:

Январь	80	Май	82	Сентябрь	85
Февраль	78	Июнь	85	Октябрь	84
Март	75	Июль	87	Ноябрь	86
Апрель	80	Август	82	Декабрь	88

Визуализируем данные, представив значения аргументов в виде векторов (рис. 9.12):

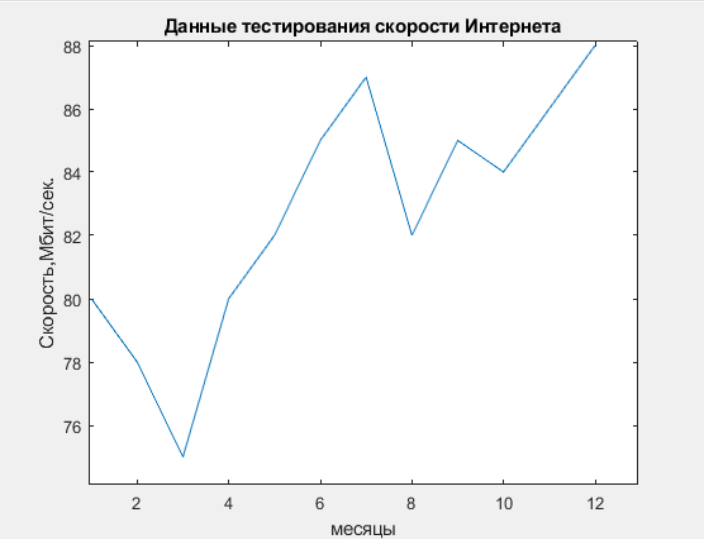


Рис. 9.12. График значений скорости Интернета

```
>> exp_1.m
month=[1 2 3 4 5 6 7 8 9 10
11 12];
speed=[80 78 75 80 82 85 87
82 85 84 86 88];
plot(month,speed, '-')
title ('Данные тестирования
скорости Интернета')
xlabel ('месяцы')
ylabel('Скорость,Мбит/сек.')
```

Визуальный анализ данных не позволяет сделать каких-либо выводов о наличии тенденции или сезонности в значениях данных ряда: в отдельные месяцы, например в феврале, марте, августе, октябре и декабре, скорость снижалась по сравнению с предыдущими месяцами, в остальные периоды – возрастала.

Применим к исходным данным метод укрупнения интервалов, образовав новый динамический ряд с более крупными временными периодами – кварталами, и рассчитаем среднюю скорость Интернета в каждом квартале, например с помощью MATLAB или Excel:

I 77,7
II 82,3
III 84,7
IV 86,0

Воспользуемся функцией **reshape ()**:

```
month=[1 2 3 4 5 6 7 8 9 10 11 12];
% кварталы
kv=[1 2 3 4];
speed=[80 78 75 80 82 85 87 82 85 84 86 88];
%разбиваем на кварталы по 3 месяца каждый
per_speed=reshape(speed,3,[]);
per_m=mean(per_speed);
```

```
per_m =
77.6667   82.3333   84.6667   86.0000
```

Визуализируем полученные данные (рис. 9.13):

```
month=[1 2 3 4 5 6 7 8 9 10 11 12];
kv=[1 2 3 4];% кварталы
speed=[80 78 75 80 82 85 87 82 85 84 86 88];
per_speed=reshape(speed,3,[]);%разбиваем на кварталы по 3 месяца
каждый
per_m=mean(per_speed);
plot(kv,per_m, '-')
title ('Данные тестирования скорости Интернета')
xlabel ('кварталы')
ylabel('Средняя скорость,Мбит/сек.')
```

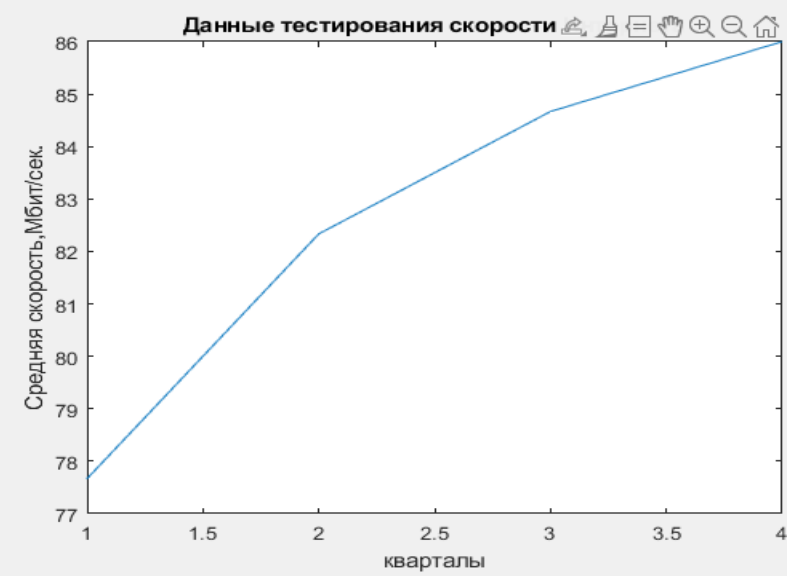


Рис. 9.13. Ежеквартальная средняя скорость Интернета

По новым, более крупным, интервалам и графику уже четко видно, что значения исследуемого признака во временном аспекте имеют тенденцию к возрастанию.

Применение рассмотренного метода в основном ограничивается теми ситуациями, когда исходные данные относятся к дням, неделям или месяцам года, так как значения исследуемого признака по более мелким временным интервалам больше подвержены случайным колебаниям. Если временные промежутки представляют собой годы, то укрупнение интервалов становится малоэффективным.

9.3.2. Метод наименьших квадратов

Применяется для решения различных математических задач, в том числе для анализа данных, и основан на минимизации суммы квадратов отклонений функций от исходных переменных. Например, когда нужно найти зависимость между значениями двух переменных, заданными выборочными данными (в этом случае речь идет об отклонениях теоретических значений данных от экспериментальных), или сделать прогноз и т. д.

Суть метода заключается в следующем. Пусть две величины x и y связаны табличной зависимостью, полученной, например, из опытов.

x_1	x_2	x_3	…	x_n
y_1	y_2	y_3	…	y_n

На плоскости xOy данной таблице соответствует n точек M_i (x_i, y_i), $i = 1, 2, 3, \ldots, n$. Точки M_i называют экспериментальными точками (рис. 9.14).

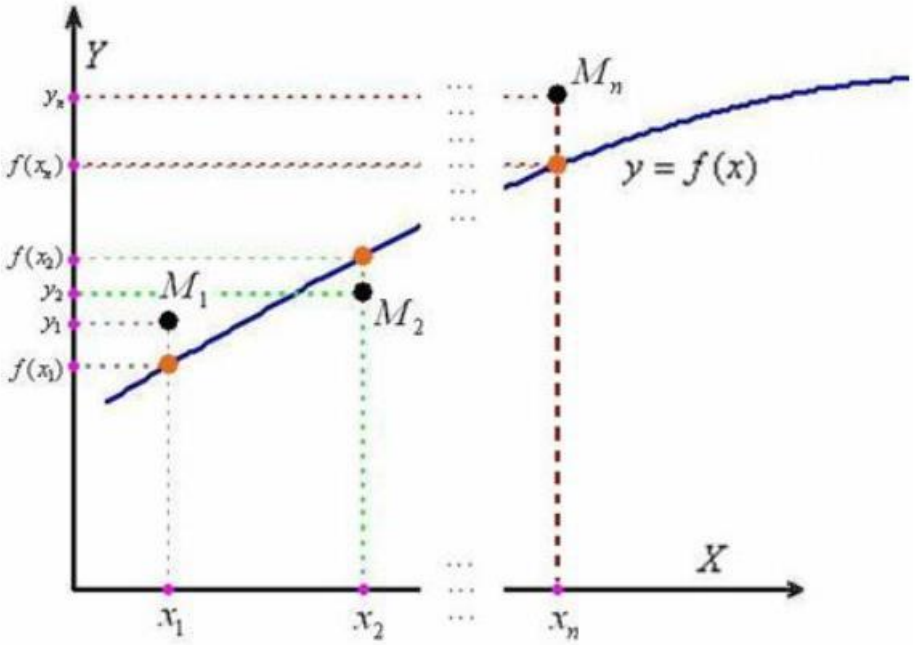


Рис. 9.14. Экспериментальные точки

Требуется установить функциональную зависимость $y = f(x)$ между переменными x и y по результатам экспериментальных исследований, приведенных в таблице. Необходимо отыскать такой метод

подбора эмпирической формулы, который позволяет не только найти саму формулу, но и оценить погрешность подгонки. В общем случае искомая функция $y = f(x)$ будет зависеть не только от x, но и от некоторого количества параметров: $y = f(a, b, c, ...)$.

Постановка задачи. Найти аппроксимирующую функцию

$$y = f(a, b, c, ...) \tag{9.1}$$

такую, чтобы в точках $x = x_i$ она принимала значения, по возможности близкие к табличным, т. е. график искомой функции должен проходить как можно ближе к экспериментальным точкам. Вид функции (9.1) может быть известен из теоретических соображений или определяться характером расположения экспериментальных точек M_i на плоскости xOy.

Для отыскания коэффициентов a, b, ... в функции (9.1) и применяется метод наименьших квадратов, который состоит в следующем. Между искомой функцией и табличными значениями в точках x_i наблюдаются отклонения. Обозначим их $\Delta y = f(x_i, a, b, ...) - y_i$, где $i = 1, 2, 3, ..., n$. Выбираем значения коэффициентов a, b, ... так, чтобы сумма квадратов отклонений принимала минимальное значение:

$$S(a, b, ...) = \sum_{i=1}^{n} (\Delta y_i)^2 = \sum_{i=1}^{n} [f(x_i, a, b, ...)^2 - y_i{}^2] \to \min. \tag{9.2}$$

Сумма $S(a, b, \ldots)$ – функция нескольких переменных.

Необходимый признак экстремума функции нескольких переменных – обращение в нуль частных производных:

$$S'_a = 0, S'_b = 0, ... \tag{9.3}$$

План решения задачи следующий.

1. Выбираем функцию $y = f(x, a, b, ...)$.

2. Для отыскания коэффициентов a, b, ... составляем систему уравнений (9.3).

3. Решая систему уравнений (9.3), находим значения коэффициентов a, b, ...

4. Подставляя a, b, ... в уравнение (9.1), получаем искомую функцию $y = f(x, a, b, ...)$.

5. По достаточному признаку экстремума функции нескольких переменных следует убедиться в постоянстве знака дифференциала второго порядка этой функции: $d^2S > 0$ при любых приращениях ар-

гументов da, db, ... Такая проверка проводится в теоретической части метода наименьших квадратов и на практике не повторяется.

6. Обычно рассматривают несколько видов функций $y = f(x, a, b, ...)$ и выбирают ту функцию, для которой суммарная погрешность $\sum_{i=1}^{n}[f(x_i, a, b, ...)^2 - y_i{}^2]$ окажется наименьшей.

Рассмотрим несколько случаев подбора аппроксимирующей функции $y = f(x, a, b, ...)$.

Линейная функция. Общий вид:

$$y = ax + b. \tag{9.4}$$

По формуле (9.2) составим функцию двух переменных и найдем, при каких значениях a, b эта функция принимает минимальное значение, учитывая, что по необходимому признаку экстремума производных [см. формулу (9.3)] частные производные функции (9.2) должны быть равны нулю:

$$\begin{cases} S'_a(a,b) = \sum_{i=1}^{n} 2(ax_i + b - y_i)x_i = 0, \\ S'_b(a,b) = \sum_{i=1}^{n} 2(ax_i + b - y_i)1 = 0. \end{cases} \tag{9.5}$$

Преобразуем уравнения системы (9.5):

$$\begin{cases} (\sum_{i=1}^{n} x_i^2)a + (\sum_{i=1}^{n} x_i^2)b = \sum_{i=1}^{n} x_i y_i. \\ (\sum_{i=1}^{n} x_i)a + nb = \sum_{i=1}^{n} y_i. \end{cases} \tag{9.5a}$$

Получаем систему линейных уравнений с двумя неизвестными a и b. Коэффициентами при a и b являются соответствующие суммы, которые вычисляются из исходной табличной зависимости и постоянны для данной выборки. При различных значениях x_i главный определитель этой системы отличен от нуля:

$$\begin{aligned} \Delta = \begin{vmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{vmatrix} = n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2 = \\ = \sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^2 \neq 0. \end{aligned} \tag{9.6}$$

Следовательно, система имеет единственное решение, вычисляемое по формуле Крамера:

$$\begin{aligned} a &= \frac{\Delta a}{\Delta} = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}, \\ b &= \frac{\Delta b}{\Delta} = \frac{1}{n}\sum_{i=1}^{n} y_i - a\frac{1}{n}\sum_{i=1}^{n} x_i. \end{aligned} \tag{9.7}$$

Подставим найденные значения a и b в уравнение (9.4) и получим исходную линейную функцию $y = ax + b$.

Убедимся, что в стационарной точке $M_0\,(a,\,b)$ функция $S\,(a,\,b)$ имеет минимум.

Достаточное условие того, что функция двух переменных принимает минимальное значение, – постоянство знака дифференциала второго порядка этой функции: $\mathrm{d}^2S\,(a,\,b) > 0$ при любых приращениях аргументов $\mathrm{d}a$, $\mathrm{d}b$.

Дифференциал второго порядка функции $S\,(a,\,b)$ имеет вид

$$\begin{gathered}\mathrm{d}^2S(a,b) = S''_{aa}\mathrm{d}a^2 + 2S''_{ab}\mathrm{d}a\mathrm{d}b + S''_{bb}\mathrm{d}b^2 = \\ = (2\sum_{i=1}^{n} x_i^2)\mathrm{d}a^2 + 2(2\sum_{i=1}^{n} x_i)\mathrm{d}a\mathrm{d}b + (2n)\mathrm{d}b^2.\end{gathered} \tag{9.8}$$

Дифференциал второго порядка представляет собой квадратичную форму 2-го порядка от переменных $\mathrm{d}a$ и $\mathrm{d}b$. Квадратичная форма принимает только положительные значения при $\mathrm{d}a \neq 0$ и $\mathrm{d}b \neq 0$, если соответствующая ей матрица положительно определена.

Матрица квадратичной формы дифференциала второго порядка (матрица Гессе) имеет вид

$$M = \begin{pmatrix} S''_{aa} & S''_{ab} \\ S''_{ab} & S''_{bb} \end{pmatrix} = \begin{pmatrix} 2\sum_{i=1}^{n} x_i^2 & 2\sum_{i=1}^{n} x_i \\ 2\sum_{i=1}^{n} x_i & 2n \end{pmatrix} \tag{9.9}$$

Найдем ее главные миноры:

$$\begin{gathered}H_1 = S''_{aa} = 2\sum_{i=1}^{n} x_i^2 > 0, \\ H_2 = \begin{vmatrix} S''_{aa} & S''_{ab} \\ S''_{ab} & S''_{bb} \end{vmatrix} = \begin{pmatrix} 2\sum_{i=1}^{n} x_i^2 & 2\sum_{i=1}^{n} x_i \\ 2\sum_{i=1}^{n} x_i & 2n \end{pmatrix} = \\ = 4n\sum_{i=1}^{n} x_i^2 - 4(\sum_{i=1}^{n} x_i)^2 = 2\sum_{i=1}^{n}\sum_{i=1}^{n}(x_i - x_j)^2 > 0.\end{gathered} \tag{9.10}$$

Так как главные миноры матрицы Гессе положительны, то по критерию Сильвестра матрица положительно определена и квадратичная форма дифференциала $\mathrm{d}^2S\,(a,\,b)$, соответствующая этой матрице, принимает только положительные значения. Из условия $\mathrm{d}^2S\,(a,\,b) > 0$ следует, что $M_0\,(a,\,b)$ – точка минимума функции $S\,(a,\,b)$.

Таким образом, коэффициенты a и b, найденные с помощью метода наименьших квадратов, всегда определяют именно минимум функции $S\,(a,\,b)$. Более того, так как функция $S\,(a,\,b)$ имеет един-

ственную стационарную точку M_0 (a, b), минимум функции – наименьшее значение S (a, b).

Если коэффициенты линейной функции найдены, можно вычислить суммарную погрешность по формуле (9.2).

Квадратичная функция. Имеет вид $y = ax^2 + bx + c$. Составим функцию трех переменных и найдем, при каких значениях a, b, c эта функция принимает минимальное значение:

$$S(a,b,c) = \sum_{i=1}^{n}(ax_i^2 + bx_i + c - y_i)^2 \to \min. \qquad (9.11)$$

Функция $S(a,b,c)$ будет принимать минимальное значение, если частные производные $S'_a(a,b,c), S'_b(a,b,c)$ и $S'_c(a,b,c)$ обращаются в нуль:

$$\begin{cases} \{S'_a(a,b,c) = \sum_{i=1}^{n} 2(ax_i^2 + bx_i + c - y_i)^2 x_i^2 = 0, \\ \{S'_b(a,b,c) = \sum_{i=1}^{n} 2(ax_i^2 + bx_i + c - y_i) \;\; x_i \;\; = 0, \\ \{S'_c(a,b,c) = \sum_{i=1}^{n} 2(ax_i^2 + bx_i + c - y_i) \;\; 1 = 0. \end{cases} \qquad (9.12)$$

Преобразуем уравнения системы следующим образом:

$$\begin{cases} (\sum_{i=1}^{n} x_i^4)a + (\sum_{i=1}^{n} x_i^3)b + (\sum_{i=1}^{n} x_i^2)c = \sum_{i=1}^{n} x_i^2\, y_i, \\ (\sum_{i=1}^{n} x_i^3)a + (\sum_{i=1}^{n} x_i^2)b + (\sum_{i=1}^{n} x_i^2)c = \sum_{i=1}^{n} x_i^1\, y_i, \\ (\sum_{i=1}^{n} x_i^2)a + (\sum_{i=1}^{n} x_i^1)b + nc = \sum_{i=1}^{n} y_i\,. \end{cases} \qquad (9.13)$$

Получили систему трех линейных уравнений с тремя неизвестными a, b, c. Аналогично случаю двух переменных эта система имеет единственное решение. Кроме того, можно доказать, что коэффициенты, найденные с помощью метода наименьших квадратов, всегда определяют именно минимум функции $S(a, b, c)$.

Решая систему уравнений, найдем значения коэффициентов a, b, c. Подставив найденные значения коэффициентов в уравнение, получим искомую квадратичную функцию. Суммарная погрешность

$$S(a,b,c) = \sum_{i=1}^{n}(ax_i^2 + bx_i + c - y_i)^2. \qquad (9.14)$$

Степенная функция. Имеет вид $y = \beta x^a$.

Прологарифмируем по основанию e функцию и получим новое уравнение $\ln y = a\ln x + \ln\beta$.

Обозначим $Y = \ln y, X = \ln x, b = \ln\beta$.

Тогда равенство примет вид $Y = aX + b$, где переменные X и Y связаны следующей табличной зависимостью.

$X = \ln x$	$X_1 = \ln x_1$	$X_2 = \ln x_2$	…	$X_n = \ln x_n$
$Y = \ln y$	$Y_1 = \ln y_1$	$Y_2 = \ln y_2$	…	$Y_n = \ln y_n$

Находим значения коэффициентов a и b. Учитывая введенные замены, получаем $\beta = e^b$. Подставив найденные значения α и β в уравнение, имеем искомую степенную функцию $y = \beta x^a$.

Суммарная погрешность

$$S(a, b) = \sum_{i=1}^{n} [\beta x_i^a - y_i]^2. \quad (9.15)$$

9.3.3. Методика расчета методом наименьших квадратов

Метод наименьших квадратов для линейной функции широко применяется при обработке данных в теории измерений, в математической статистике при нахождении статистических оценок параметров и построении уравнения линейной регрессии, в эконометрике при нахождении трендов, а также в других прикладных дисциплинах.

Рассмотрим пример задания. Экспериментальные данные о значениях переменных *x* и *y* приведены ниже.

x_i	1	2	4	6	8
y_i	3	2	1	0,5	0

Используя метод наименьших квадратов, аппроксимируем эти данные несколькими видами зависимостей: линейной и квадратичной. По результатам аппроксимации определим вид функции, лучше выравнивающей экспериментальные данные.

Вычислим параметры *a* и *b* линейной функции $y = ax + b$.

Параметры a и b уравнения $y = ax + b$ по методу наименьших квадратов можно найти из системы уравнений (9.6):

$$\begin{cases} a \sum x_i^2 + b \sum x_i = \sum x_i y_i , \\ a \sum x_i + bn = \sum y_i , \end{cases} \quad (9.16)$$

где суммирование ведется по i от 1 до n, n = 5.

Составим расчетную таблицу.

						сумма
x_i	1	2	4	6	8	21
y_i	3	2	1	0,5	0	6,5
x_i^2	1	4	16	36	64	121
$x_i y_i$	3	4	4	3	0	14

Получаем систему

$$\begin{cases} 121a + 21b = 14, \\ 21a + 5b = 6{,}5. \end{cases} \quad (9.17)$$

Откуда аналитически по формуле (9.6), или применив функцию **polyfit ()** в MATLAB, находим a = –0,405, b = 3,003, т. е. получаем линейную функцию y = –0,405x + 3,003.

Соответствующий скрипт для линейной функции в MATLAB будет выглядеть так:

```
x=[1 2 4 6 8];
y=[3 2 1 0.5 0];
%суммирование элементов массива
s_x=sum (x,'all');
s_y=sum(y,'all');
%почленное вычисление значения x_i^2
p_x=x.^2;
%вычисление значения ∑x_i^2
summa=sum(p_x);% );
%вычисление x_i · y_i
p=x.*y;
%вычисление значения ∑x_i · y_i
summa_x_y=sum(p);
% вычисляем коэффициенты a и b
pol = polyfit(x, y, 1);
%вывод результатов вычислений на экран
disp(x);
disp(y);
disp(p_x);
disp(summa);
disp (s_x);
disp (s_y);
disp (summa_x_y);
disp (pol);
```

```
    1     2     4     6     8
3.0000    2.0000    1.0000    0.5000         0
    1     4    16    36    64
       121
        21
        6.5000
        14
    -0.4055    3.0030
```

Для выяснения, какой вид функции лучше описывает тренд в контексте метода наименьших квадратов, вычислим сумму квадратов отклонений для линейной функции:

```
x=[1 2 4 6 8];
y=[3 2 1 0.5 0];
%суммирование элементов массива
s_x=sum (x,'all');
s_y=sum(y,'all');
%почленное возведение в квадрат элементов массива
p_x=x.^2;
%суммирование квадратов элементов массива x
summa=sum(p_x);
p=x.*y;
summa_x_y=sum(p);
%вычисляем коэф. a b
pol = polyfit(x, y, 1);

%уравнение линии
func=-0.4055*x+3.003;
%разность квадратов отклонений
y1=(y-func).^2;
%сумма разности квадратов отклонений
M=sum (y1);
%вывод результатов вычислений на экран
disp ('значения y');
disp(y);
disp ('значения линейной функции');
disp(func);
disp('разность квадратов отклонений');
```

```
disp(y1);
disp ('сумма разности квадратов отклонений');
disp (M);
значения y
   3.0000    2.0000    1.0000    0.5000         0
значения линейной функции
   2.5975    2.1920    1.3810    0.5700   -0.2410
разность квадратов отклонений
   0.1620    0.0369    0.1452    0.0049    0.0581
значение суммы разности квадратов отклонений
   0.4070
```

Вычислим сумму квадратов отклонений для квадратичной функции $y = ax^2 + bx + c$.

Параметры a, b и c уравнения по методу наименьших квадратов можно найти из системы уравнений (9.11).

Составим расчетную таблицу.

						сумма
x_i	1	2	4	6	8	21
y_i	3	2	1	0,5	0	6,5
x_i^4	1	16	256	1296	4096	5665
x_i^3	1	8	64	216	512	801
x_i^2	1	4	16	36	64	121
$x_i \cdot y_i$	3	4	4	3	0	14

```
x=[1 2 4 6 8];
y=[3 2 1 0.5 0];
%суммирование элементов массива
s_x=sum (x,'all');
s_y=sum(y,'all');
%почленное возведение в четвертую степень элементов массива
p4_x=x.^4;
%почленное возведение в третью степень элементов массива
p3_x=x.^3;
%почленное возведение в квадрат элементов массива
p2_x=x.^2;
%произведение x_i^2 · y_i
p=p2_x.*y;
```

```
%суммирование четвертых степеней элементов массива x
summa4=sum(p4_x);
%суммирование третьих степеней элементов массива x
summa3=sum(p3_x);
%суммирование квадратов элементов массива x
summa2=sum(p2_x);
summa_x_y=sum(p);
%вычисляем коэф. a b c
pol = polyfit(x, y, 2);
%вывод результатов вычислений на экран
disp(x);
disp ('значения y');
disp(y);
disp ('сумма элементов массива x');
disp (s_x);
disp ('сумма элементов массива y');
disp (s_y);
disp ('p4_x, p3_x p2_x, summa');
disp(p4_x);
disp(p3_x)
disp(p2_x);
disp (p);
disp (summa4);
disp (summa3);
disp(summa2);
disp(summa_x_y);
disp ('коэффициенты a b c');
disp (pol);
>> exp_4_2
    1     2     4     6     8
значения y
   3.0000    2.0000    1.0000    0.5000         0
сумма элементов массива x
   21
сумма элементов массива y
```

```
   6.5000
p4_x, p3_x p2_x, summa
1        16        256       1296       4096
    1    8    64   216   512
    1    4    16    36    64
    3    4     4     3     0
      5665
  801
  121
   14
```

Коэффициенты a b c

```
   0.0522   -0.8718    3.6975
```

Тогда система уравнений (9.7) примет вид

$$\begin{cases} 5665a + 21b + 121c = 45, \\ 801a + 121b + 21c = 14\ , \\ 121a + 21b + 5c = 6{,}5. \end{cases} \qquad (9.18)$$

Таким образом, уравнение квадратичной функции имеет вид

$$y = 0{,}0522x^2 - 0{,}8718x + 3{,}6975. \qquad (9.19)$$

Вычислим сумму разности квадратов отклонений для квадратичной функции:

```
>> exp_4_2
x=[1 2 4 6 8];
y=[3 2 1 0.5 0];
%суммирование элементов массива
s_x=sum (x,'all');
s_y=sum(y,'all');
%почленное возведение в четвертую степень элементов массива
p4_x=x.^4;
%почленное возведение в третью степень элементов массива
p3_x=x.^3;
%почленное возведение в квадрат элементов массива
p2_x=x.^2;
%произведение элементов массивов x и y
p=x.*y;
```

```
%суммирование четвертых степеней элементов массива x
summa4=sum(p4_x);
%суммирование третьих степеней элементов массива x
summa3=sum(p3_x);
%суммирование квадратов элементов массива x
summa2=sum(p2_x);
summa_x_y=sum(p);
%вычисляем коэф. a b c
pol = polyfit(x, y, 2);
func=0.0522*x.^2-0.8718*x+3.6975;
%разность квадратов отклонений
y1=(y-func).^2;
%сумма разности квадратов отклонений
M=sum (y1);
%вывод результатов вычислений на экран
disp(x);
disp ('значения y');
disp(y);
disp ('сумма элементов массива x');
disp (s_x);
disp ('сумма элементов массива y');
disp (s_y);
disp ('p4_x, p3_x p2_x, summa');
disp(p4_x);
disp(p3_x)
disp(p2_x);
disp (p);
disp (summa4);
disp (summa3);
disp(summa2);
disp(summa_x_y);
disp ('коэффициенты a b c');
disp (pol);
disp('разность квадратов отклонений');
disp(y1);
```

```
disp ('сумма разности квадратов отклонений');
disp (M);
разность квадратов отклонений
   0.0149    0.0265    0.0021    0.0237    0.0041
сумма разности квадратов отклонений
   0.0713
```

Так как значение суммы разности квадратов отклонений, найденное с помощью линейной функции (0,4070), значительно больше соответствующего значения для квадратичной функции (0,0713), можно сделать вывод, что квадратичная функция лучше в контексте метода наименьших квадратов выравнивает данные.

Визуализируем экспериментальные данные. Добавим к ним графики линейной и квадратичной функций. Соответствующие скрипт и график представлены ниже.

```
>> kv_plt
n=5;
x=[1 2 4 6 8];
y=[3 2 1 0.5 0];
%суммирование элементов массива
s_x=sum (x,'all');
s_y=sum(y,'all');
%почленное возведение в квадрат элементов массива
p_x=x.^2;
%суммирование квадратов элементов массива x
summa=sum(p_x);
p=x.*y;
summa_x_y=sum(p);
%вычисляем коэф. a b
pol = polyfit(x, y, 1);
% уравнение линии
func=-0.4055*x+3.003;
%разность квадратов отклонений
y1=(y-func).^2;
%сумма разности квадратов отклонений
M=sum (y1);
```

```
% квадратичная функция
%почленное возведение в четвертую степень элементов массива
p4_x=x.^4;
%почленное возведение в третью степень элементов массива
p3_x=x.^3;
%почленное возведение в квадрат элементов массива
p2_x=x.^2;
%произведение элементов массивов x и y
p_v=p2_x.*y;
%суммирование четвертых степеней элементов массива x
summa4=sum(p4_x);
%суммирование третьих степеней элементов массива x
summa3=sum(p3_x);
%суммирование квадратов элементов массива x
summa2=sum(p2_x);
summa_x_y=sum(p_v);
% вычисляем коэф. a b c
pol = polyfit(x, y, 2);
func_2=0.0522*x.^2-0.8718*x+3.6975;
% разность квадратов отклонений
y1_2=(y-func_2).^2;
%сумма разности квадратов отклонений
M_2=sum (y1_2);
%вывод экспериментальных данных (vd)графиков линейной (lf) и
квадратичной (qf)функций
plot(x, y, '.', x,func, 'R-',x,func_2, 'B--');
title ('Экспериментальные данные');
xlabel ('значения x');
ylabel ('значения y');
text(x,func,'lf'); %подпись для графика линейной функции
text(x,func_2,'qf');% подпись для графика квадратичной функции
text(x,y,'vd');%подпись для экспериментальных данных x y
```

На рис. 9.15 представлены совмещенные экспериментальные данные.

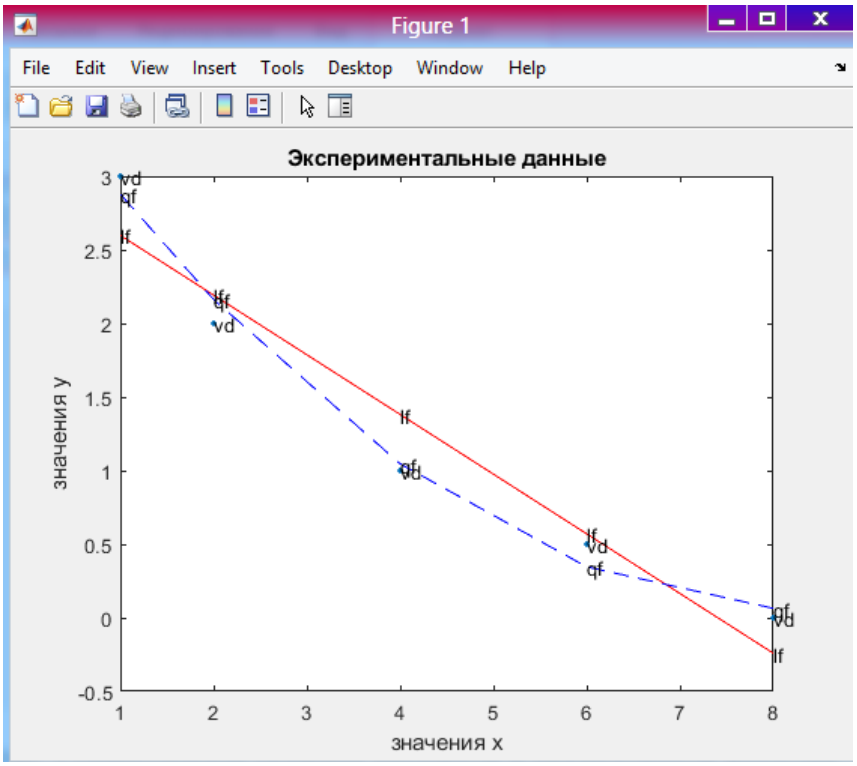


Рис. 9.15. Совмещенные экспериментальные данные.
Графики линейной и квадратичной функции

Визуализация данных подтверждает сделанный ранее вывод о том, что квадратичная функция лучше линейной функции в контексте метода наименьших квадратов выравнивает данные.

Аналогично вычисления выполняются для степенной функции.

9.3.4. Метод скользящей средней. Методика расчета методом скользящей средней

Метод основан на расчете и анализе так называемых скользящих (подвижных) средних. *Скользящими (подвижными) средними* называют средние арифметические значения показателя, исчисленные по новым *m*-членным укрупненным интервалам.

Правила построения этих интервалов следующие. Первый из интервалов включает в себя первые *m* уровней ряда динамики. Второй интервал образуется путем исключения первого члена укрупненного интервала и замены его последующим элементом ряда динамики, имеющим номер (*m* + 1), и так далее до включения в интервал последнего уровня ряда. По вычисленным подобным путем подвижным средним делают вывод о существовании тенденции в динамическом ряду.

Наиболее часто на практике используются линейные фильтры. Общая формула линейного фильтра следующая:

$$Y(t) = \sum_{r=-1}^{k} a_r X(t+r), \qquad (9.20)$$

где $Y(t)$ – сглаженное (отфильтрованное) значение временного ряда в момент времени t; a_r – вес, приписываемый значению исходного ряда, находящемуся на расстоянии r от рассматриваемого момента времени t.

Фильтр [см. формулу (9.20)] учитывает k значений (уровней) ряда после момента времени t и l уровней до него. Число $k + 1 + l$ значений исходного ряда, одновременно участвующих в сглаживании, называется *шириной интервала сглаживания*. Если $k = l$, то сглаживание центрированное.

Сглаженный ряд короче исходного ряда на $k + 1$ значение. В зависимости от выбора ширины интервала сглаживания, величины a_r применяются различные методы сглаживания.

Если $\sum a_r = 1$ и $a_r = \text{const}$, то фильтр [см. формулу (9.20)] отражает вычисление среднего арифметического, которое называют скользящей средней.

Для удобства сопоставления сглаженного и исходного рядов в качестве ширины интервала сглаживания чаще выбирают нечетное число $m = 2k + 1$. Тогда $a_r = \frac{1}{m}$ и получаем

$$Y(t) = \frac{1}{2k+1} \sum_{r=-k}^{k} (t+r). \qquad (9.21)$$

Такое сглаживание будет симметричным ($a_{-r}=a_r$) и центрированным. Чаще всего для сглаживания берут m = 3; 5; 7.

Если дисперсия уровней исходного ряда постоянная и равна σ^2, а сами члены ряда $X(t_i)$ независимы между собой, то дисперсия сглаженного ряда $Y(t)$ равна $\frac{\sigma^2}{m}$. Таким образом, при увеличении колеблемости исходного ряда $X(t)$ (при наличии большой дисперсии σ^2) для уменьшения амплитуды колебаний у сглаженного ряда $Y(t)$ необходимо увеличивать ширину интервала сглаживания m либо проводить процедуру сглаживания повторно.

По степени уменьшения дисперсии у повторно сглаженных рядов можно судить о степени зависимости между собой членов исходного ряда, т. е. о наличии «долговременной памяти» у исходного ряда.

Если ряд имеет периодические колебания с продолжительностью цикла меньше m, то они полностью исчезают при сглаживании с помощью скользящей средней с шириной интервала сглаживания m.

Расчет $Y(t)$ при $m > 3$ можно упростить, применяя рекуррентную формулу

$$Y(t) = Y(t-1) + \frac{X(t+k) - X(t+k-1)}{2k+1}, t = k+2; k+3; \ldots; m-k. \quad (9.22)$$

При расчетах первое сглаженное значение $Y(k+1)$ вычисляется по формуле (9.21):

$Y(k+1) = (X(1) + X(2) + \ldots + X(m))/m$, $m = 2k+1$.

Недостаток метода простой скользящей средней – равное участие в сглаживании значений ряда, отстоящих от момента сглаживания t на разном расстоянии. В результате этого могут быть потеряны важные для анализа свойства ряда.

Применение метода скользящей средней рассмотрим на основе данных тестирования скорости Интернета, полученных сервисом сайта https://2ip.ru/speed и представленных раннее (см. п. 9.3.1).

Проведем сглаживание ряда методом скользящей средней по трем членам (табл. 9.1).

Если в качестве укрупненного интервала используют период в три месяца, то первая подвижная трехчленная средняя вычисляется как среднее арифметическое из данных за январь, февраль и март, вторая – как среднее арифметическое из данных за февраль, март, апрель и т. д. Значения подвижных средних относят к конкретному временному периоду, соответствующему середине укрупненного интервала.

Таблица 9.1

Сглаживание ряда динамики методом скользящей средней по трем членам

Исходные данные		Расчетные данные	
Месяц	Скорость, Мбит/с	Скользящая сумма трех членов	Скользящая средняя по трем членам (расчетные уровни ряда) y_t
Январь	80	–	–
Февраль	78	80 + 78 + 75= 233	$\frac{233}{3} = 77{,}7$
Март	75	78 + 75 + 80 = 233	$\frac{233}{3} = 77{,}7$

Окончание табл. 9.1

Исходные данные		Расчетные данные	
Месяц	Скорость, Мбит/с	Скользящая сумма трех членов	Скользящая средняя по трем членам (расчетные уровни ряда) y_t
Апрель	80	75 + 80 + 82 = 237	$\frac{237}{3} = 79{,}0$
Май	82	80 + 82 + 85 = 247	$\frac{247}{3} = 82{,}3$
Июнь	85	82 + 85 + 87 = 254	$\frac{254}{3} = 84{,}7$
Июль	87	85 + 87 + 82 = 254	$\frac{254}{3} = 84{,}7$
Август	82	85 + 87 + 82 = 254	$\frac{254}{3} = 84{,}7$
Сентябрь	85	82 + 85 + 84 = 251	$\frac{251}{3} = 83{,}7$
Октябрь	84	85 + 84 + 86 = 255	$\frac{255}{3} = 85{,}0$
Ноябрь	86	84 + 86 + 88 = 258	$\frac{258}{3} = 86{,}0$
Декабрь	88	–	–

В отличие от метода укрупнения интервалов, ряд данных разрезаем на «скользящие» кусочки по три месяца в каждом, т. е. первая скользящая средняя относится к периоду январь – февраль – март, вторая – к периоду февраль – март – апрель, третья – к периоду март – апрель – май и так далее, последняя – к периоду октябрь – ноябрь – декабрь. Скользящая средняя для периода ноябрь – декабрь не вычисляется.

```
N=12;
L=10;
month=[1 2 3 4 5 6 7 8 9 10 11 12];
speed=[80 78 75 80 82 85 87 82 85 84 86 88];
s=size(L);
sum=0;
k=1;
for i=1:L
  for j=i:i+2
     sum=sum+speed (j);
  end
  s(k)=round (sum/3,1);
```

```
  k=k+1;
  sum=0;
end
disp('средние значения по кварталам');
disp (s);
>> scol_avg1
средние значения по кварталам
   77.7000   77.7000   79.0000   82.3000   84.7000   84.7000   84.7000
83.7000   85.0000   86.0000
```

Визуализируем данные (рис. 9.16).

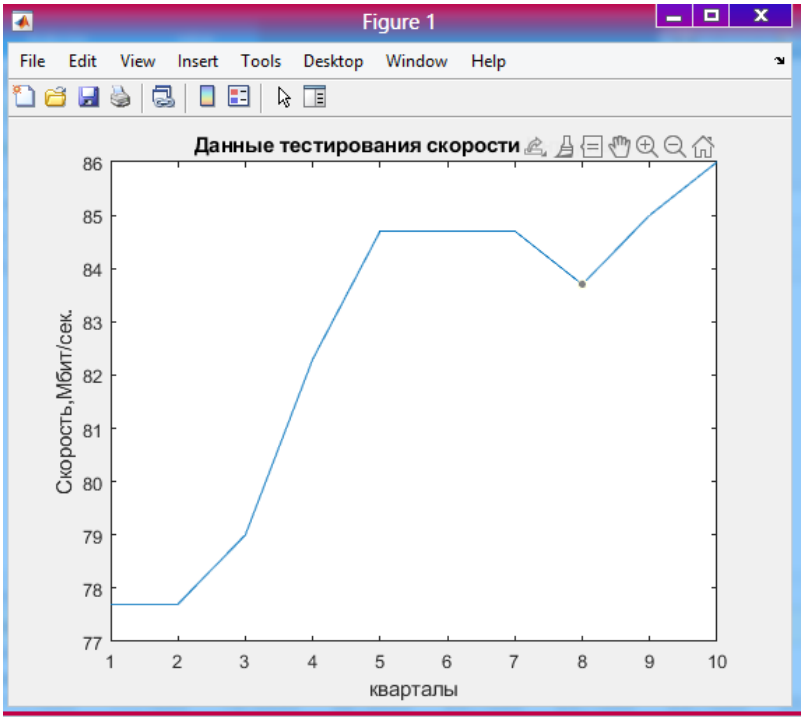


Рис. 9.16. Данные тестирования скорости Интернета, представленные графически

Сравнивая с графиком, полученным методом укрупнения интервалов, видим, что для этой задачи метод скользящей средней позволяет, помимо линейного тренда, выявить сезонные составляющие – период с мая по июнь, для которых характерна постоянная скорость Интернета; в июле наблюдаем уменьшение скорости. Для большей достоверности результатов необходимо анализировать данные не за один год, как в рассматриваемом примере, а за период в три года и более.

9.3.5. Метод центрирования скользящих средних. Методика расчета

Методика расчета центрированных скользящих средних показана ниже (табл. 9.2).

Таблица 9.2

Сглаживание ряда динамики методом скользящей средней по четырем членам

Исходные данные		Расчетные данные	
Месяц	Скорость, Мбит/с	Нецентрированные скользящие средние по четырем членам, Мбит/с	Центрированные скользящие средние по четырем членам, Мбит/с
Январь	80	–	–
Февраль	78	–	–
Март	75	$\frac{80+78+75+80}{4}=78{,}3$	$\frac{78{,}3+78{,}8}{2}=78{,}6$
Апрель	80	$\frac{78+75+80+82}{4}=78{,}8$	$\frac{78{,}8+80{,}5}{2}=79{,}7$
Май	82	$\frac{75+80+82+85}{4}=80{,}5$	
Июнь	85	$\frac{80+82+85+87}{4}=83{,}5$	$\frac{80{,}5+83{,}5}{2}=82{,}0$
Июль	87	$\frac{82+85+87+82}{4}=84{,}0$	$\frac{83{,}5+84{,}0}{2}=83{,}8$
Август	82	$\frac{85+87+82+85}{4}=84{,}8$	$\frac{84{,}0+84{,}8}{2}=84{,}4$
Сентябрь	85	$\frac{87+82+85+84}{4}=84{,}5$	$\frac{84{,}8+84{,}4}{2}=84{,}6$
Октябрь	84	$\frac{82+85+84+86}{4}=84{,}3$	$\frac{84{,}5+84{,}3}{2}=84{,}4$
Ноябрь	86	$\frac{85+84+86+88}{4}=85{,}8$	$\frac{84{,}3+85{,}8}{2}=85{,}1$
		–	–
Декабрь	88	–	–

В тех случаях, когда сглаживание проводится по четному числу уровней ряда динамики, середина временного интервала сглаживания будет находиться между двумя моментами (периодами) времени. Например, если проводить сглаживание по четырем членам, середина первого интервала будет находиться между февралем и мартом, второго интервала – между мартом и апрелем и т. д. В таких случаях возникает необходимость центрирования полученных результатов для отнесения сглаженных значений показателя к конкретным периодам или моментам времени. Расчет центрированных скользящих средних может проводиться в два этапа:

1) определение скользящих сумм и нецентрированных скользящих средних по четному числу уровней ряда динамики;

2) исчисление центрированных скользящих средних из двух смежных, ранее исчисленных нецентрированных скользящих средних и отнесение их к соответствующим периодам или моментам времени.

Соответствующие скрипт и график (рис. 9.17) в MATLAB приведены ниже.

```
N=12;
L=10;
month=[1 2 3 4 5 6 7 8 9 10 11 12];
speed=[80 78 75 80 82 85 87 82 85 84 86 88];
s=size(L); %создаем пустой массив
sum=0;
k=1;
for i=1:L-1
  for j=i:i+3
     sum=sum+speed (j);
  end
  s(k)=round (sum/4,1);
  k=k+1;
  sum=0;
end
%размер массива для центрированных скользящих средних
M=k-1;
s1=size(M);
k=1;
for i=1:M-2
```

```
   for j=i:i+1
      sum=round ((s(j)+s(j+1))/2,1);
   end
  s1(k)=sum;
  k=k+1;
 end
disp('нецентрированные средние значения по 4 месяцам');
disp (s);
disp('центрированные средние значения по 4 месяцам');
disp (s1);
>> center_avg
нецентрированные средние значения по 4 месяцам
78.3000    78.8000    80.5000    83.5000    84.0000    84.8000    84.5000
84.3000   85.8000
центрированные средние значения по 4 месяцам
78.5000  79.7000    82.0000    83.8000    84.4000    84.7000    84.4000
85.1000
```

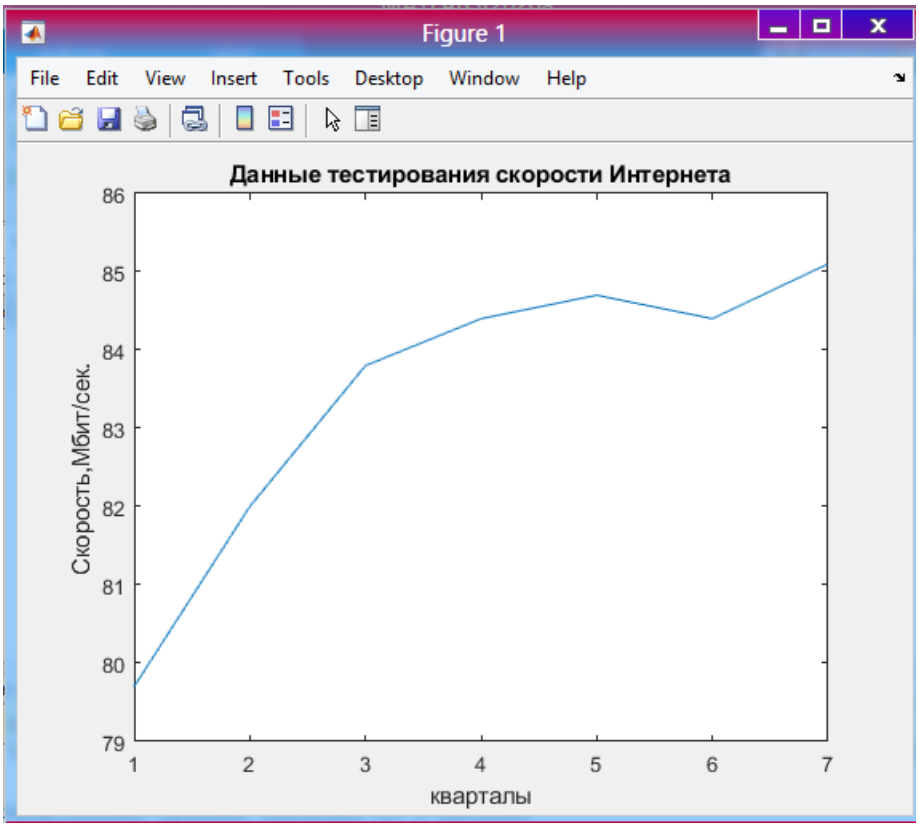


Рис. 9.17. Данные о скорости Интернета с использованием метода центрирования скользящих средних

Данные, полученные методом центрирования скользящих средних, и их визуализация позволяют сделать вывод о более высоком качестве сглаживания по сравнению с методом скользящих средних. Таким образом, данный метод лучше выявляет тенденцию в рассматриваемом ряду данных.

9.3.6. Методы взвешенных скользящих средних

Суть методов взвешенных скользящих средних заключается в том, что значениям исходного ряда приписывается вес a_r, зависящий от расстояния до середины интервала сглаживания, т. е. от $|r|$. Тогда $a - r = a_r$ и сглаживание по этим методам центрированное и симметричное. Для определения значений веса прибегают к различным подходам.

Рассмотрим *первый подход.* Пусть значениями веса являются члены разложения бинома $(0{,}5 + 0{,}5)^{2k}$. Ширина интервала определяется формулой $m = 2k + 1$. Тогда

$$a_r = C_{2k}^{k-r}(0{,}5)^{k-r}\,(0{,}5)^{2k-(k-r)}, r = 0, 1, \dots, k, a_r = C_{2k}^{k-r}(0{,}5)^{2k}, \tag{9.23}$$

где C_{2k}^{k-r} – число сочетаний из $2k$ элементов по $k - r$ элементов. При этом $a_{-r} = C_{2k}^{k-(-r)} = C_{2k}^{k+r}$.

По свойству сочетаний имеем

$$C_{2k}^{k-r} = C_{2k}^{2k-(k-r)} = C_{2k}^{k+r} = \frac{(2k)!}{(2k-k-r)!(k+r)!} = \frac{(2k)!}{(k-r)!(k+r)!}.$$

Получаем:

– при $m = 3$ ($k = 1$) $a_{-1} = 1/4$, $a_0 = 1/2$, $a_1 = 1/4$;

– при $m = 5$ ($k = 2$) $a_{-2} = 1/16$, $a_{-1} = 1/4$, $a_{-0} = 3/8$, $a_1 = 1/4$, $a_2 = 1/16$;

– при $m = 7$ ($k = 3$) $a_{-3} = 1/64$, $a_{-2} = 3/32$, $a_{-1} = 15/64$,

$a_{-0} = 5/16$, $a_1 = 15/64$, $a_2 = 3/32$, $a_3 = 1/64$.

Второй подход заключается в подборе полинома регрессии к данным, содержащимся в интервале сглаживания. При этом свободный член a полинома регрессии выбирается равным расчетному значению ряда $Y(t)$.

Рассмотрим случай, когда уравнение регрессии квадратичное, т. е. сглаживание происходит на основе уравнения параболы. В этом случае для каждого набора m последовательных членов исходного

ряда составляется система $m = 2k + 1$ уравнений для расчета методом наименьших квадратов:

$$Z(t+i) = a + b_i + c_i^2; I = -k;\ -k+1;\ -1; 0; 1; \ldots; k-1; k,$$

где $Z(t + i)$ – расчетные значения квадратичного уравнения регрессии.

Сглаженное значение ряда $Y(t)$ выбирается по формуле

$$Y(t) = Z(t + 0) = a. \tag{9.24}$$

Запишем систему нормальных уравнений для определения коэффициентов параболы a, b, c:

$$\left\{\begin{array}{c} am + b\sum_{i=-k}^{k} i + c\sum_{i=-k}^{k} i^2 = \sum_{i=-k}^{k} X(t+i) \\ a\sum_{i=-k}^{k} i + b\sum_{i=-k}^{k} i^2 + c\sum_{i=-k}^{k} i^3 = \sum_{i=-k}^{k} iX(t+i) \\ a\sum_{i=-k}^{k} i^2 + b\sum_{i=-k}^{k} i^3 + c\sum_{i=-k}^{k} i^4 = \sum_{i=-k}^{k} i^2X(t+i) \end{array}\right\} \tag{9.25}$$

Так как $\sum_{i=-k}^{k} i = \sum_{i=-k}^{k} i^3$, получаем из системы (9.25) систему

$$\left\{\begin{array}{c} am + c\sum_{i=-k}^{k} i^2 = \sum_{i=-k}^{k} X(t+i) \\ b\sum_{i=-k}^{k} i^2 = \sum_{i=-k}^{k} iX(t+i) \\ a\sum_{i=-k}^{k} i^2 + c\sum_{i=-k}^{k} i^4 = \sum_{i=-k}^{k} i^2X(t+i) \end{array}\right\} \tag{9.25а}$$

Исключив c из первого и третьего уравнений системы (9.25а), получаем формулу для расчета коэффициента a:

$$Y(t) = a\frac{\sum_{i=-k}^{k} i^4 \sum_{i=-k}^{k} X(t+i) - \sum_{i=-k}^{k} i^2 \sum_{i=-k}^{k} (i^2X(t+i))}{m\sum_{i=-k}^{k} i^4 - \sum_{i=-k}^{k} i^2}. \tag{9.26}$$

При $m = 5$, $k = 2$ имеем

$a = -(3/35)X(t - 2) + (128/35)X(t - 1) + (17/35)X(t) + (12/35)X(t + 1) -$
$- (3/35)X(t + 2)$,

т. е. $a_r = -3/35; 128/35, 17/35, 12/35, -3/35$.

Аналогично для $m = 7$, $k = 3$ и для $m = 9$, $k = 4$ соответственно получаем

$a_r = -2/21, 3/21, 6/21, 7/21, 6/21, 3/21, -2/21$;

$a_r = -21/231, 14/231, 39/231, 54/231, 39/231, 14/231, -21/231$.

Можно легко проверить, что если в качестве сглаживающего многочлена взять прямую, то коэффициенты $a_r = 1/m$ совпадут с коэффициентами сглаживания, полученными с помощью метода простой скользящей средней.

Предположим, дисперсия уровней исходного ряда постоянная и равна δ^2, а сами члены ряда $X(t_i)$ независимы между собой. В этом случае дисперсия сглаженного по квадратичному полиному ряда $Y(t)$

$$\delta_y^2 = \frac{3(3k^2+3k-1)}{(2k-1)(2k+1)(2k+3)}\delta^2, \text{ при } m = 5, k = 2\ \delta_y^2 = \frac{17}{35}\delta^2, .$$

а при $m = 7$, $k = 3$ $\delta_y^2 = \frac{1}{3}\delta^2$, т. е. тенденция к уменьшению дисперсии с ростом m сохраняется.

При сглаживании с помощью скользящей средней нет возможности получить сглаженные значения для k первых и k последних членов ряда $X(t)$. В случае сглаживания с помощью полинома регрессии для крайних членов исходного ряда могут быть получены сглаженные значения – значения полинома регрессии в этих точках. Но для этого надо оценить не только свободный член a полинома, но и остальные коэффициенты полинома регрессии (коэффициенты b, c в квадратичном случае). Из системы (9.25) получаем

$$b = \frac{\sum_{i=-k}^{k} iX(y+i)}{\sum_{i=-k}^{k} i^2}, c = \frac{\sum_{i=-k}^{k} X(y+i) - am}{\sum_{i=-k}^{k} i^2}.$$

Для $m = 5$, $k = 2$ для начальных значений ряда имеем

$a = Z(0) = Y(0) = Y(3) = -(3/35)X(1) + (12/35)X(2) + (17/35)X(3) +$
$+ (12/35)X(4) - (3/35)X(5)$, $b = 1/10(-2X(1) - X(2) + X(4) + 2X(5))$,

$c = 1/10(X(1) + X(2) + X(3) + X(4) + X(5) - 5a)$.

$Y(1) = Z(-2) = a + b(-2) + c(-2)2 = a - 2b - 4c$,

$Y(2) = Z(-1) = a + b(-1) + c(-1)2 = a - b - 2c$.

Для последних пяти членов ряда аналогично получаем (n – объем выборки):

$a = Z(0) = Y(n-2) = -(3/35)X(n-4) + (12/35)X(n-3) +$
$+ (17/35)X(n-2) + (12/35)X(n-1) - (3/35)X(n)$,

$b = 1/10\ (-2X(n-4) - X(n-3) + X(n-1) + 2X(n))$,

$c = 1/10\ (X(n-4) + X(n-3) + X(n-2) + X(n-1) + X(n) - 5a)$,

$Y(n-1) = Z(1) = a + b(1) + c(1)2 = a + b + 2c$, $Y(n) = Z(2) =$
$= a + b(2) + c(2)2 = a + 2b + 4c$.

В практике технического анализа наибольшее распространение получили линейно-взвешенное скользящее среднее (ЛВСС) и экспоненциально-взвешенное скользящее среднее (ЭВСС).

Формула расчета ЛВСС в общем виде выглядит следующим образом:

$$WMA_t = \frac{nP_t+(n-1)P_{t-1}+(n-2)P_{t-2}+\cdots+1\,P_{t-(n-1)}}{n+(n-1)+(n-2)+\cdots+2+1}, \qquad (9.27)$$

где n – интервал сглаживания; P_{t-i} – значение показателя в период времени $(t - i)$.

Знаменатель представляет собой арифметическую прогрессию и для удобства расчетов может быть преобразован следующим образом:

$$\sum_{i=1} i = \frac{n(n-1)}{2}, \quad \sum_{i=1}^{n} i = \frac{n(n-1)}{2}. \qquad (9.28)$$

Таким образом, формулу (9.27) можно представить так:

$$WMA_t = \frac{2\sum_{i=0}^{n-1}(n-i)P_{t-i}}{n(n-1)}. \qquad (9.29)$$

9.3.7. Методика расчета методами взвешенных скользящих средних

Рассмотрим методику расчета линейно-взвешенного скользящего среднего на примере данных о котировках акций, представленных в табл. 9.3.

Таблица 9.3

Данные о котировках акций

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P	5,3	6,7	7,9	7,1	5,2	4,1	3,5	5,4	7,3	9,4	8,0	6,6	7,9	9,2	7,6
EMA	–	–	–	–	6,5	5,7	4,8	4,7	5,5	6,9	7,6	7,5	7,7	8,2	8,0

Предположим, что интервал сглаживания равен пяти. В этом случае первое значение *WMA* может быть рассчитано для 5-го периода. Подставив имеющиеся данные в формулу (9.29), получим значение *WMA*, равное 9,68.

$$WMA_5 = \frac{2(5 \cdot 5{,}2 + 4 \cdot 7{,}1 + 3 \cdot 7{,}9 + 2 \cdot 6{,}7 + 1 \cdot 5{,}3)}{5(5-1)}.$$

Следующее значение *WMA* составит уже 8,51:

$$WMA_6 = \frac{2(5 \cdot 4{,}1 + 4 \cdot 5{,}2 + 3 \cdot 7{,}1 + 2 \cdot 7{,}9 + 1 \cdot 6{,}7)}{5(5-1)}.$$

Дальнейшие расчеты проводятся аналогично, а их результаты представлены на графике (рис. 9.18).

Рис. 9.18. Визуализация линейно-взвешенного скользящего среднего

Преимущество этого показателя перед простым скользящим средним – меньшее запаздывание. Это происходит в силу того, что наиболее старые данные имеют незначительный весовой коэффициент, а следовательно, направление тренда устанавливается главным образом по последним данным. Например, если интервал сглаживания равен 15, то удельный вес трех последних значений цен будет равен 0,4, а первых трех 0,06.

(15 + 14 + 13)/105* = 0,4

(3 + 2 + 1)/105 = 0,06

Вопросы для самоконтроля

1. В чем суть метода укрупнения интервалов? Продемонстрируйте на примере.

*105 представляет собой сумму чисел от 1 до 15.

2. В чем суть метода наименьших квадратов? Продемонстрируйте на примере.

3. Приведите формулу расчета суммы квадратов отклонений.

4. Приведите формулы расчета коэффициентов эмпирического парного линейного уравнения регрессии по методу наименьших квадратов.

5. Приведите формулы расчета коэффициентов эмпирического парного квадратичного уравнения регрессии по методу наименьших квадратов.

6. Приведите формулы расчета коэффициентов эмпирического парного степенного уравнения регрессии по методу наименьших квадратов.

7. По экспериментальным данным, приведенным в тексте (п. 9.3.1), вычислите сумму разности квадратов отклонений для степенной функции. Сделайте вывод.

8. В чем суть метода скользящих средних? Чем этот метод отличается от метода центрирования скользящих средних? Продемонстрируйте на примере.

9. В чем суть методов взвешенных скользящих средних? Каково их отличие от методов скользящей средней (центрирования скользящей средней)? Продемонстрируйте на примере.

Практическая работа

МЕТОДЫ СГЛАЖИВАНИЯ ВРЕМЕННОГО РЯДА

Задание. Требуется сгладить временной ряд методами:

– укрупнения интервалов;

– простой скользящей средней;

– центрированной скользящей средней;

– взвешенной средней с выбором значений веса:

– в разложении бинома;

– из уравнения квадратичной регрессии.

Отчет по практической работе должен содержать:

– титульный лист (см. приложение);

– содержание;

– для каждого метода:

○ описание используемого метода;

○ алгебраический расчет по описанному методу;

○ скрипт, соответствующий выполненным расчетам;

○ график, на котором точками отмечены исходные данные, линией – сглаженные данные;

– общий график, на котором точками отмечены исходные данные, а линиями – данные, сглаженные четырьмя методами;

– вывод по динамике данных;

– вывод на основе визуализации данных о том, какой метод позволяет лучше выявить динамику в рассматриваемом ряду данных.

Варианты заданий

Вариант 1. Задан ряд показателей урожайности зерновых культур в целом по России (ц/га) за 1984 – 1998 гг.: 15,6; 17,6; 16,4; 15,6; 17,6; 20,3; 15,8; 18,8; 17,9; 15,6; 12,5; 14,0; 17,8; 10,4; 10,6.

Вариант 2. Дана таблица значений некоторой функциональной зависимости, полученной из $n = 10$ опытов.

x	1	2	3	4	5	6	7	8	9	10
y	1,0	1,5	3,0	4,5	7,0	8,5	8,0	11,3	1,0	9,5

Вариант 3. В таблице ниже приведены данные о среднедушевом прожиточном минимуме в день одного трудоспособного x (руб.) и среднедневной заработной плате y (руб.) по 12 регионам России.

Номер региона	1	2	3	4	5	6	7	8	9	10	11	12
x, руб.	78	82	87	79	89	106	67	88	73	87	76	115
y, руб.	133	148	134	154	162	195	139	158	152	162	159	173

Вариант 4. Сделайте прогноз курса валют на завтра. В качестве базы расчета используйте данные Центрального банка РФ за предыдущие две недели по российскому рублю.

Вариант 5. В таблице ниже представлен список наиболее интересных моделей оперативной памяти, указаны размер памяти (Гб), цена.

Модель оперативной памяти	Размер, Гб	Цена, руб.
Kingston ValueRAM KVR16N11/8	16	2820
Samsung M378A1K43CB2-CTD	8	2880
ПК HyperX Fury HX432C16FB3K2/8	16	3480
Corsair CMSA4GX3M1A1066C7	4	1789
Crucial CT8G4DFS824A	8	2885

Вариант 6. В таблице представлены данные зависимости тока от мощности и частоты вращения двигателя.

Мощность, кВт	Среднее значение токов холостого хода (в долях от силы номинального тока) при синхронной частоте вращения, об/мин				
	3000	1500	1000	750	600
0,5 – 1	0,4	0,55	0,6	–	–
1,1 – 5	0,35	0,5	0,55	0,6	–
5,1 – 10	0,25	0,45	0,5	0,55	0,6
10,1 – 25	0,2	0,4	0,45	0,5	0,55
25,1 – 50	0,18	0,35	0,4	0,45	0,5

Вариант 7. Данные рейтинга субъектов Федерации по средней выручке за 2019 г. приведены с сайта https://www.spark-interfax.ru/statistics.

Субъект Федерации	Средняя выручка, млн руб.
Москва	117,17
Санкт-Петербург	100,19
Московская область	82,03
Тюменская область	156,35
Свердловская область	63,43
Краснодарский край	57, 85
Республика Татарстан	54,55
Нижегородская область	71,60
Самарская область	55,52
Республика Башкортостан	59,41

Вариант 8. Сделайте прогноз курса валют на завтра. В качестве базы расчета используйте данные Центрального банка РФ за предыдущие две недели по воне (Южная Корея).

Вариант 9. Данные рейтинга субъектов Федерации по количеству юридических лиц за 2019 г. приведены с сайта https://www.spark-interfax.ru/statistics.

Субъект Федерации	Количество компаний, шт.
Москва	711 017
Санкт-Петербург	279 544
Московская область	206 395
Тюменская область	80 213
Свердловская область	126 322
Краснодарский край	116 358
Республика Татарстан	107 415
Нижегородская область	81 270
Самарская область	97 637
Новосибирская область	102 527

Вариант 10. Данные рейтинга субъектов Федерации по количеству компаний в области энергетики за 2019 г. приведены с сайта https://www.spark-interfax.ru/statistics.

Субъект Федерации	Количество компаний, шт.
Москва	1500
Санкт-Петербург	741
Московская область	1190
Тюменская область	509
Свердловская область	700
Краснодарский край	534
Челябинская область	548
Нижегородская область	543
Красноярский край	534
Новосибирская область	587

Вариант 11. Данные о численности населения и количестве трудоспособных за период с 2006 по 2019 г. приведены с сайта https://rosinfostat.ru/ekonomicheski-aktivnoe-naselenie.

Год	Всего, тыс. чел.	Занятые
2006	74 419	69 169
2007	75 289	70 770
2008	75 700	71 003
2009	75 694	69 410
2010	75 478	69 934
2011	75 779	70 857
2012	75 676	71 545
2013	75 529	71 391
2014	75 428	71 539
2015	76 588	72 324
2016	76 636	72 393
2017	76 285	72 316
2018	76 190	72 532
2019	75 398	71 933

Вариант 12. Сделайте прогноз курса валют на завтра. В качестве базы расчета используйте данные Центрального банка РФ за предыдущие две недели по белорусскому рублю.

Вариант 13. Статистика естественного движения населения по данным Росстата приведена с сайта https://rosinfostat.ru/smertnost/#i-2.

Наименование субъекта	Родившихся	Умерших
Российская Федерация	1 604 344	1 828 910
Центральный федеральный округ	391 129	508 436
Северо-Западный федеральный округ	145 537	176 127
Южный федеральный округ	173 257	210 304
Северо-Кавказский федеральный округ	141 841	73 338
Приволжский федеральный округ	311 450	390 946
Уральский федеральный округ	147 057	146 947
Сибирский федеральный округ	196 185	224 041
Дальневосточный федеральный округ	97 888	98 721

Вариант 14. Сделайте прогноз курса валют на завтра. В качестве базы расчета используйте данные Центрального банка РФ за предыдущие две недели по евро.

Вариант 15. Сделайте прогноз стоимости оперативной памяти размером 8 Гб в зависимости от ее модели.

Модель оперативной памяти	Размер, Гб	Цена, руб.
G.SKILL Ripjaws V F4-3200C16D-16GVKB	16	6600
HyperX Fury HX316C10FB/8	8	3772
Kingston ValueRAM KVR26S19S8/8	8	2600
AMD R538G1601S2S-UO	8	2490
G.SKILL Trident Z RGB F4-3200C14D-32GTZR	32	22 740
Samsung M378A4G43MB1-CTD	32	10 858
Thermaltake TOUGHRAM RGB R009D408GX2-4400C19A	16	14 210
Crucial CT8G4DFS824A	8	2600

Вариант 16. Сделайте прогноз курса валют на завтра. В качестве базы расчета используйте данные Центрального банка РФ за предыдущие две недели по американскому доллару.

ЗАКЛЮЧЕНИЕ

В результате изучения материалов пособия будущие специалисты смогут использовать основные приемы обработки и представления экспериментальных данных, осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий, прогнозировать тренды.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Полный список стандартных функций MATLAB [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/functionlist.html (дата обращения: 21.09.2022).

2. Метод наименьших квадратов : метод. указания / сост. Л. В. Коломиец, Н. Ю. Поникарова. – Самара : Изд-во Самар. ун-та, 2017. – 32 с.

3. Создание mat-файла в MATLAB [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=CDpW82MPA2o&t=20s (дата обращения: 21.09.2022).

4. Как установить инструмент Insert в MATLAB [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=lyEPDSwf9dk&t=22s (дата обращения: 21.09.2022).

5. Тип datetime [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/ref/datetime.html (дата обращения: 21.09.2022).

6. Форматы файлов и функции импорта и экспорта [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/import_export/supported-file-formats.html (дата обращения: 21.09.2022).

7. Обработка недостающих данных [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/data_analysis/missing-data-in-matlab.html (дата обращения: 21.09.2022).

8. Задача cleandata [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/data_analysis/cleandatawithliveeditortasks.html (дата обращения: 21.09.2022).

9. Типы данных в MATLAB [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/data-types.html (дата обращения: 21.09.2022).

10. Работа со строками [Электронный ресурс]. – Режим доступа: https://exponenta.ru/news/rabota-so-strokami-v-MATLAB (дата обращения: 21.09.2022).

11. Импорт данных в MATLAB [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/ref/importdata.html (дата обращения: 21.09.2022).

12. Описание числового типа данных [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/matlab_prog/integers.html (дата обращения: 21.09.2022).

13. Описание символьного типа данных [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/ref/char.html (дата обращения: 21.09.2022).

14. Управление внешним видом и поведением Line-объекта [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/ref/matlab.graphics.chart.primitive.line-properties.html (дата обращения: 21.09.2022).

15. Документация по работе с функцией plot [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/ref/plot.html (дата обращения: 21.09.2022).

16. Представление дат и времени [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/referencelist.html?type=function&category=date-and-time-operations (дата обращения: 21.09.2022).

17. Работа с данными типа «таблица» [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/tables.html (дата обращения: 21.09.2022).

18. Работа с данными типа «расписания» [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/referencelist.html?type=function&category=timetables (дата обращения: 21.09.2022).

19. Базовые функции [Электронный ресурс]. – Режим доступа: https://hub.exponenta.ru/post/kniga-spravochnik-po-matlab-matematicheskie-funktsii-vgpotemkin600#abs (дата обращения: 21.09.2022).

20. Массивы ячеек [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/cell-arrays.html (дата обращения: 21.09.2022).

21. Работа с массивами различных типов [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=8TUxIRpMj7E&feature=emb_logo (дата обращения: 21.09.2022).

22. Временные ряды [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/ref/timeseries.html (дата обращения: 21.09.2022).

23. Примеры создания временных рядов [Электронный ресурс]. – Режим доступа: https://exponenta.ru/events/2y10jfvh2vk3ebdbzblfig4ohnrcbr8hv3cv6tpgybt2wqbg05ci (дата обращения: 21.09.2022).

24. Массивы структур [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/ref/struct.html (дата обращения: 21.09.2022).

25. Функция strrep [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/ref/strrep.html (дата обращения: 21.09.2022).

26. Построение линейного тренда в MATLAB [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=dOADS2jpgD4&feature=emb_logo (дата обращения: 21.09.2022).

27. Вычисление среднего значения [Электронный ресурс]. – Режим доступа: https://docs.exponenta.ru/matlab/ref/mean.html (дата обращения: 21.09.2022).

28. Работа со строками в MATLAB [Электронный ресурс]. – Режим доступа: https://exponenta.ru/news/rabota-so-strokami-v-MATLAB (дата обращения: 21.09.2022).

29. Математические функции [Электронный ресурс]. – Режим доступа: http://old.exponenta.ru/soft/Matlab/potemkin/book2/chapter6/contens.asp (дата обращения: 21.09.2022).

ПРИЛОЖЕНИЕ

Образец оформления титульного листа к практическим работам

Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

_____

название подразделения

ОТЧЕТ
К ПРАКТИЧЕСКОЙ РАБОТЕ …
по дисциплине
«_____»

название дисциплины

Выполнил:
ст. гр. _____

название группы

_____

ФИО

Принял:

_____

должность, ФИО преподавателя

Владимир 202_

ОГЛАВЛЕНИЕ

Учебное издание

АРТЮШИНА Лариса Андреевна
ТРОИЦКАЯ Елена Анатольевна

АНАЛИЗ ДАННЫХ В СРЕДЕ MATLAB

Учебно-практическое пособие

Редактор Т. В. Евстюничева
Технические редакторы Ш. Ш. Амирсейидов, О. В. Балашова
Компьютерная верстка Е. А. Герасиной
Выпускающий редактор А. А. Амирсейидова

Подписано в печать 22.12.22.
Формат 60×84/16. Усл. печ. л. 15,81. Тираж 40 экз.
Заказ
Издательство
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых.
600000, Владимир, ул. Горького, 87.