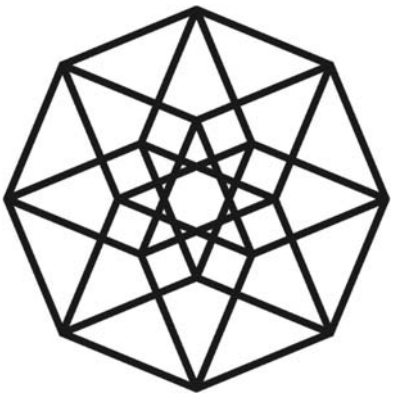
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Высшая математика»

# СОВРЕМЕННЫЕ МАТЕМАТИЧЕСКИЕ СИСТЕМЫ

***Методические рекомендации к лабораторным работам для студентов направления подготовки 01.03.04 «Прикладная математика» очной формы обучения***



Могилев 2021

УДК 519.67
ББК 22.195.5
М21

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Высшая математика» «27» мая 2021 г., протокол № 9

Составители: ст. преподаватель О. А. Маковецкая;
доц. И. И. Маковецкий

Рецензент канд. техн. наук, доц. В. М. Ковальчук

Методические рекомендации разработаны на основе рабочей программы по дисциплине «Современные математические системы» для студентов направления подготовки 01.03.04 «Прикладная математика» и предназначены для использования при выполнении лабораторных работ по дисциплине в третьем семестре.

Учебно-методическое издание

СОВРЕМЕННЫЕ МАТЕМАТИЧЕСКИЕ СИСТЕМЫ

| | |
|---|---|
| Ответственный за выпуск | В. Г. Замураев |
| Корректор | И. В. Голубцова |
| Компьютерная верстка | Н. П. Полевничая |

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 56 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

# Содержание

# Введение

Целью изучения дисциплины «Современные математические системы» является формирование у студентов навыков работы с математическими пакетами, позволяющими автоматизировать математические вычисления.

В третьем семестре, в соответствии с рабочей программой дисциплины, студенты выполняют 17 лабораторных работ, варианты и условия которых приведены в методических рекомендациях.

К защите каждой лабораторной работы студент подготавливает отчёт, который оформляется с использованием текстовых редакторов и включает: название и цель работы, формулировку задания. Лабораторные работы выполняются с применением Mathcad PTC 6.0, GNU Octave, пакета R.

# 1 Лабораторная работа № 1. Создание документа, ввод формул, программирование в Mathcad, графическая система

## *1.1 Теоретические сведения*

PTC Mathcad Prime применяется для выполнения, документирования и совместного использования вычислений и инженерных расчетов. PTC Mathcad Prime позволяет объединить инженерные математические записи, форматированные тексты, графики и изображения в единый документ, что облегчает визуализацию, проверку и документирование знаний и совместную работу. При вводе уравнений в документ PTC Mathcad Prime автоматически рассчитывает результаты. PTC Mathcad Prime отличается надежностью и обладает всеми функциональными возможностями, необходимыми для вычислений, обработки данных и инженерных расчетов.

PTC Mathcad позволяет брать производные, рассчитывать корни уравнений, анализировать данные, решать системы уравнений и обыкновенные дифференциальные уравнения. При изменении любого числа, переменной или уравнения PTC Mathcad Prime выполняет мгновенный перерасчет всех данных в документе.

PTC Mathcad Prime позволяет легко выполнять преобразование единиц измерения из одной системы в другую и помогает находить ошибки в единицах измерения, используемых в вычислениях. Единицы измерения поддерживаются в функциях, графиках и массивах.

Сразу после запуска Mathcad откроет пустой документ. Для того чтобы создать, открыть или сохранить документ, необходимо использовать кнопку M на главной панели, в открывшемся меню выбрать «Создать документ».

В документах PTC Mathcad выполняются и отображаются вычисления. В открывшемся документе можно начинать вводить уравнения, текстовые и другие элементы в любом месте документа. В документах можно добавлять и настраивать нижние и верхние колонтитулы, а также изменять отображение сетки и полей страницы.

Чтобы ввести математические и другие элементы, необходимо предварительно задать область. Области документа содержат данные элементов. Существуют следующие типы областей:

– Math;
– таблица;
– график;
– блок решения;
– изображение;
– текстовое поле;
– блок текста.

Можно открыть одновременно несколько документов. Каждый открытый документ помещается на новую вкладку под лентой. Чтобы перейти в нужный

документ, щелкните его вкладку. Чтобы закрыть документ, нажмите клавиши Ctrl + W.

Математические выражения в документе PTC Mathcad Prime отображаются в привычной форме: деление с дробной чертой, показатели степени в верхнем положении и составные дроби. Редактор уравнений PTC Mathcad Prime работает внутри структуры математического выражения. При вводе PTC Mathcad Prime собирает части выражения, используя правила приоритета и дополнительные правила для упрощения ввода знаменателей, степеней и подкоренных выражений.

Часть математических выражений можно выбрать с помощью клавиши пробела или клавиш со стрелками. Математические элементы можно группировать по одному либо все одновременно. Расположение курсора с левой или правой стороны элемента может повлиять на группировку. Выбранную группу можно копировать и вставлять в другую область формул. К выбранной группе можно применять оператор до тех пор, пока группа не будет включать в себя результат.

Группа, не включающая результат, подобна набору элементов, заключенному в невидимые круглые скобки. Например, операнды представляют собой группы, поэтому при замене или удалении оператора группы остаются без изменений.

Вводить математические выражения в PTC Mathcad так же просто, как писать на листе бумаги. Выражения можно вводить прямо с клавиатуры, используя обычные математические обозначения.

В область формул или текстовую область можно вводить следующие типы математических выражений:

– алгебраические;
– дифференциальные и интегральные;
– логические;
– матрицы и векторы;
– таблицы;
– строковые.

PTC Mathcad вычисляет математические выражения в порядке слева направо, сверху вниз. При изменении математического выражения его результат автоматически обновляется. Все связанные математические области, находящиеся справа или снизу от измененных выражений, соответственно обновляются. Если отключить расчет области, результаты не будут обновляться.

Для повторяющегося вычисления выражения или функции можно использовать переменную-диапазон или вектор. Для сохранения результаты можно записывать в файл или копировать и вставлять.

В PTC Mathcad существует четыре принципиально разных типа знака равенства:

1) оператор «определение» – определяет переменные и функции;
2) оператор «вычисление» – вычисляет выражение;

3) оператор «равно» – сравнивает переменные, векторы или выражения. Этот оператор используется в логических выражениях или ограничениях для блоков решений;

4) локальный оператор определения переменной-диапазона – определяет переменную-диапазон для оператора суммы или произведения.

Все переменные, которые используются в выражении слева от оператора оценки, должны быть определены. Чтобы выяснить происхождение ошибок, можно провести их трассировку. Можно настроить отображение результатов для повышения точности или изменения числового формата. Также можно изменить единицы измерения результата.

С помощью программы можно группировать расчеты и возвращать только определенные результаты. В PTC Mathcad создание программ осуществляется с помощью оператора «программа». Этот оператор является многошаговым контейнером для управляющих операторов PTC Mathcad, которые осуществляют выполнение следующих функций:

– назначение математических выражений локальным переменным или функциям;

– проверка условий ветвления;

– выполнение расчетов в цикле;

– добавление точек останова;

– отслеживание ошибок.

Используя совместно функции PTC Mathcad и операторы программирования, можно составлять довольно сложные программы. Например, можно составить программу для выполнения следующих действий:

– введение математических выражений в естественной математической нотации;

– вставка единиц измерения;

– вызов встроенных констант и функций;

– обращение к определенным ранее переменным и функциям;

– использование логических операторов совместно с условными операторами;

– использование операторов для работы с массивами, например оператор элемента в матрице, совместно с переменными итерации;

– определение нескольких результатов при обработке матрицы или вложенной матрицы.

PTC Mathcad вычисляет операторы в программе в порядке, который определяют операторы программирования. PTC Mathcad возвращает результат на последнем шаге.

Например, выходные данные следующей программы зависят от последнего шага или указанного возвращенного значения.

Программы используются для определения выражений, которые иначе были бы неоднозначны, трудночитаемы, или таких, для которых необходимо использовать диапазоны, условные функции и массивы. Программы также используются в целях экономии памяти – промежуточные вычисления над большими массивами выполняются внутри тела программы. Промежуточные

значения хранятся в памяти только во время выполнения программы, иначе они занимали бы память до закрытия всего документа.

Можно создавать законченные программы, программы, состоящие из одного шага, без ввода оператора программы. Можно вставлять программы в вычисляемые математические выражения. Можно вставлять программы в определения переменных и функций. Переменную или функцию можно затем использовать в документе.

Чтобы программы получались понятными и компактными, рекомендуется использовать условные операторы, операторы цикла и другие операторы программирования.

Программы можно использовать в блоках решения или вне их.

В PTC Mathcad можно создать следующие типы графиков:

– графики XY;

– полярные графики;

– контурные графики;

– 3D-графики.

При построении графика он первоначально отображается в виде линейной кривой. Вид графика можно изменить с помощью параметров, имеющихся на вкладке **Графики (Plots)**. Чтобы открыть список **Символ (Symbol)** или список **Стиль линии (Line Style)** в группе **Стили (Styles)**, щелкните стрелку справа от списка **Символ (Symbol)** или **Стиль линии (Line Style)** соответственно. Чтобы открыть список **Изменить тип (Change Type)** в группе **Кривые (Traces)**, выберите пункт **Изменить тип (Change Type)**.

### *Контрольные вопросы*

1 Для чего служит пакет PTC Mathcad?

2 Перечислите основные типы блоков Mathcad.

3 Какие режимы существуют для знака равенства?

4 Перечислите возможности программирования в Mathcad.

5 Какие виды графиков можно строить в Mathcad?

### *1.2 Задания для самостоятельного выполнения*

Используя возможности программирования Mathcad, создайте функции $f(x)$, $r(\varphi)$, $z(x, y)$ и постройте их графики (двумерный, в полярной системе координат, в трехмерном пространстве):

$$f(x)=\begin{cases} ax^2+bx+c,\ x\le m, \\ \dfrac{\sin\left(\sqrt[b]{x}\right)}{e^{ax^2+bx}},\ m<x\le n, \\ \dfrac{c\ln x}{\sqrt{bx^2+c}},\ x>n; \end{cases}$$

$$r(\varphi)=\frac{p\cos\varphi+q\sin\varphi}{\sqrt[b]{\cos\varphi}}+\sqrt{p\sin\varphi+q\cos\varphi};$$

$$z=\frac{(x-a)^2}{p}+\frac{(y-b)^2}{q}-c\sqrt{ax^2+by^2}.$$

Задание выполняется по вариантам в соответствии с номером студента по журналу.

1 $a = 1$, $b = 2$, c = 1, $p = 1$, $q = 2$, $m = 2$, $n = 6$.
2 $a = 1$, $b = 2$, $c = 2$, $p = 1$, $q = 3$, $m = 2$, $n = 7$.
3 $a = 1$, $b = 2$, $c = 1$, $p = 1$, $q = 4$, $m = 2$, $n = 8$.
4 $a = 1$, $b = 2$, $c = 2$, $p = 1$, $q = 5$, $m = 3$, $n = 6$.
5 $a = 1$, $b = 2$, $c = 3$, $p = 1$, $q = 6$, $m = 3$, $n = 7$.
6 $a = 1$, $b = 2$, $c = 4$, $p = 2$, $q = 1$, $m = 3$, $n = 9$.
7 $a = 1$, $b = 2$, $c = 5$, $p = 2$, $q = 2$, $m = 1$, $n = 5$.
8 $a = 1$, $b = 2$, $c = 6$, $p = 2$, $q = 3$, $m = 1$, $n = 10$.
9 $a = 1$, $b = 2$, $c = 7$, $p = 2$, $q = 4$, $m = 1$, $n = 8$.
10 $a = 1$, $b = 2$, $c = 3$, $p = 2$, $q = 5$, $m = 2$, $n = 9$.
11 $a = 1$, $b = 2$, $c = 4$, $p = 2$, $q = 6$, $m = 1$, $n = 9$.
12 $a = 1$, $b = 2$, $c = 5$, $p = 2$, $q = 7$, $m = 5$, $n = 11$.
13 $a = 1$, $b = 2$, $c = 6$, $p = 2$, $q = 8$, $m = 5$, $n = 6$.
14 $a = 1$, $b = 2$, $c = 7$, $p = 3$, $q = 2$, $m = 5$, $n = 8$.
15 $a = 1$, $b = 2$, $c = 8$, $p = 3$, $q = 3$, $m = 5$, $n = 9$.
16 $a = 1$, $b = 3$, $c = 1$, $p = 3$, $q = 4$, $m = 5$, $n = 10$.
17 $a = 1$, $b = 3$, $c = 2$, $p = 3$, $q = 5$, $m = 4$, $n = 6$.
18 $a = 1$, $b = 3$, $c = 3$, $p = 3$, $q = 6$, $m = 4$, $n = 7$.
19 $a = 1$, $b = 4$, $c = 1$, $p = 3$, $q = 7$, $m = 4$, $n = 8$.
20 $a = 2$, $b = 2$, $c = 1$, $p = 3$, $q = 8$, $m = 4$, $n = 9$.
21 $a = 2$, $b = 2$, $c = 3$, $p = 4$, $q = 2$, $m = 4$, $n = 10$.
22 $a = 2$, $b = 2$, $c = 4$, $p = 4$, $q = 3$, $m = 4$, $n = 11$.
23 $a = 2$, $b = 2$, $c = 5$, $p = 4$, $q = 4$, $m = 4$, $n = 12$.
24 $a = 2$, $b = 3$, $c = 1$, $p = 4$, $q = 5$, $m = 1$, $n = 2$.
25 $a = 2$, $b = 3$, $c = 4$, $p = 4$, $q = 6$, $m = 1$, $n = 3$.
26 $a = 2$, $b = 4$, $c = 1$, $p = 4$, $q = 7$, $m = 1$, $n = 4$.
27 $a = 3$, $b = 2$, $c = 1$, $p = 4$, $q = 8$, $m = 1$, $n = 6$.
28 $a = 3$, $b = 2$, $c = 4$, $p = 5$, $q = 3$, $m = 2$, $n = 3$.
29 $a = 3$, $b = 3$, $c = 1$, $p = 5$, $q = 4$, $m = 2$, $n = 4$.
30 $a = 3$, $b = 3$, $c = 2$, $p = 5$, $q = 5$, $m = 2$, $n = 5$.

# 2 Лабораторная работа № 2. Алгебраические вычисления в Mathcad

## *2.1 Теоретические сведения*

Mathcad позволяет выполнять различные символьные вычисления с применением оператора аналитического преобразования (вызывается одновременным нажатием кнопок Ctrl и <.>), ключевых слов и модификаторов. При выполнении оператора аналитического преобразования Mathcad упрощает выражение, к которому этот оператор применяется, при этом с помощью ключевых слов можно изменять получаемый результат в зависимости от решаемой задачи.

Ключевые слова Mathcad:

– **assume** – делает предположения относительно области определения переменных;

– **coeffs** – возвращает коэффициенты полинома в виде вектора;

– **collect** – приводит подобные слагаемые;

– **combine** – комбинирует элементы выражения с использованием свойств элементарных функций;

– **confrac** – вычисляет разложение числа или функции в непрерывную дробь;

– **expand** – выполняет перемножение всех множителей в выражении;

– **explicit** – возвращает выражения с подставленными значениями переменных, но без сокращения числовых выражений;

– **factor** – факторизация выражения (разложение в произведение);

– **float** – возвращает результаты, рассчитанные вычислениями с плавающей запятой;

– **fully** – возвращает подробное решение уравнения (выводит все возможные корни);

– **parfrac** – выполняет разложение рационального выражения в сумму дробей со знаменателями первой и второй степеней;

– **rectangular** – возвращает результаты с комплексными значениями, разделенными на действительную и мнимую части;

– **rewrite** – переписывает выражение через элементарные функции;

– **series** – выполняет разложение функции или выражения в ряд Тейлора или ряд Лорана в окрестности точки 0;

– **simplify** – упрощает алгебраически или вычисляет выражение;

– **solve** – решает уравнение аналитически;

– **substitute** – заменяет все вхождения одной переменной другой переменной, выражением или числом;

– **using** – заменяет созданную переменную в решении уравнением;

– **fourier**, **invfourier** – прямое и обратное преобразования Фурье;

– **laplace**, **invlaplace** – прямое и обратное преобразования Лапласа;

– **ztrans**, **invztrans** – прямое и обратное z-преобразования.

Ключевые слова могут использоваться с модификаторами:

– **acos**, **acot**, **asin**, **atan**, **cos**, **cosh**, **cot**, **coth**, **exp**, **gamma**, **ln**, **log**, **signum**, **sin**, **sincos**, **tan**, **tanh** – используются с ключевым словом **rewrite**, выражение будет переписано с использованием соответствующей функции;

– **ALL** – используется с ключевыми словами **assume**, **explicit**, применяет ключевое слово к каждой переменной в выражении;

– **cauchy** – возвращает главное значение Коши для интеграла;

– **complex** – применяется с **assume**, **factor**, **parfrac**, указывает, что переменная является комплексной или операция выполняется над комплексными числами;

– **degree** – используется с **coeffs**, возвращает второй столбец в выходных данных, который содержит значения в градусах;

– **domain** – используется с **factor**, **parfrac**, указывает область или набор входных значений переменной;

– **even**, **odd** – используются с **assume**, указывают на то, что значение переменной является четным или нечетным числом;

– **fraction** – используется с **confrac**, возвращает непрерывную дробь в виде дроби;

– **integer** – используется с **assume**, указывает, что значение переменной является целым числом;

– **matrix** – используется с **confrac**, возвращает непрерывную дробь в виде массива;

– **max** – используется с **simplify**, выполняет дополнительные шаги для большего упрощения;

– **raw** – используется с ключевыми словами преобразований, возвращает результат в неупрощенном виде;

– **real** – используется с **assume**, **factor**, указывает, что значение переменной является действительным числом;

– **RealRange** – используется с **assume**, указывает, что значение переменной выбирается из некоторого действительного интервала.

Ключевые слова **fully**, **using** можно использовать в качестве автономных ключевых слов без добавления предшествующего ключевого слова.

### *Контрольные вопросы*

1 Каким образом выполняются символьные алгебраические преобразования в Mathcad?

2 Чем ключевые слова отличаются от модификаторов?

3 Для чего служит ключевое слово simplify?

4 С помощью какого ключевого слова можно найти решения уравнения?

5 Каким образом можно выполнить подстановку в выражении?

6 Каким модификатором необходимо дополнить ключевое слово **assume**, чтобы указать Mathcad, что переменная в выражении является четным числом?

### *2.2 Задания для самостоятельного выполнения*

Примените к указанному рационально-тригонометрическому выражению универсальную тригонометрическую подстановку, полученный результат представьте в виде суммы дробей. Вариант задания соответствует номеру в журнале группы.

$$\frac{p\cos^{\alpha}\varphi + q\sin^{\beta}\varphi + r}{m\cos^{\beta}\varphi + n\sin^{\alpha}\varphi + k}.$$

1 $p = 1$, $q = 1$, $r = 1$, $m = 1$, $n = 2$, $k = 2$, $\alpha = 1$, $\beta = 1$.
2 $p = 1$, $q = 1$, $r = 1$, $m = 1$, $n = 2$, $k = 3$, $\alpha = 1$, $\beta = 1$.
3 $p = 1$, $q = 1$, $r = 2$, $m = 2$, $n = 2$, $k = 3$, $\alpha = 1$, $\beta = 1$.
4 $p = 1$, $q = 1$, $r = 3$, $m = 2$, $n = 1$, $k = 3$, $\alpha = 1$, $\beta = 1$.
5 $p = 2$, $q = 1$, $r = 2$, $m = 3$, $n = 2$, $k = 1$, $\alpha = 1$, $\beta = 1$.
6 $p = 2$, $q = 2$, $r = 3$, $m = 2$, $n = 1$, $k = 3$, $\alpha = 1$, $\beta = 1$.
7 $p = 2$, $q = 3$, $r = 3$, $m = 2$, $n = 1$, $k = 3$, $\alpha = 1$, $\beta = 1$.
8 $p = 4$, $q = 2$, $r = 3$, $m = 2$, $n = 1$, $k = 3$, $\alpha = 1$, $\beta = 1$.
9 $p = 1$, $q = 2$, $r = 3$, $m = 2$, $n = 3$, $k = 5$, $\alpha = 1$, $\beta = 1$.
10 $p = 1$, $q = 3$, $r = 3$, $m = 2$, $n = 2$, $k = 3$, $\alpha = 1$, $\beta = 1$.
11 $p = 1$, $q = 4$, $r = 3$, $m = 2$, $n = 3$, $k = 3$, $\alpha = 1$, $\beta = 1$.
12 $p = 3$, $q = 1$, $r = 5$, $m = 2$, $n = 1$, $k = 3$, $\alpha = 1$, $\beta = 1$.
13 $p = 2$, $q = 2$, $r = 3$, $m = 3$, $n = 1$, $k = 3$, $\alpha = 1$, $\beta = 1$.
14 $p = 1$, $q = 1$, $r = 3$, $m = 2$, $n = 1$, $k = 3$, $\alpha = 2$, $\beta = 1$.
15 $p = 1$, $q = 1$, $r = 3$, $m = 2$, $n = 1$, $k = 3$, $\alpha = 1$, $\beta = 2$.
16 $p = 1$, $q = 2$, $r = 3$, $m = 2$, $n = 2$, $k = 3$, $\alpha = 2$, $\beta = 1$.
17 $p = 2$, $q = 1$, $r = 3$, $m = 2$, $n = 1$, $k = 4$, $\alpha = 1$, $\beta = 2$.
18 $p = 3$, $q = 2$, $r = 3$, $m = 3$, $n = 1$, $k = 3$, $\alpha = 2$, $\beta = 1$.
19 $p = 1$, $q = 1$, $r = 5$, $m = 3$, $n = 2$, $k = 2$, $\alpha = 1$, $\beta = 2$.
20 $p = 1$, $q = 5$, $r = 3$, $m = 2$, $n = 1$, $k = 3$, $\alpha = 2$, $\beta = 1$.
21 $p = 5$, $q = 1$, $r = 4$, $m = 2$, $n = 1$, $k = 4$, $\alpha = 1$, $\beta = 2$.
22 $p = 1$, $q = 3$, $r = 4$, $m = 2$, $n = 3$, $k = 3$, $\alpha = 2$, $\beta = 1$.
23 $p = 2$, $q = 4$, $r = 1$, $m = 2$, $n = 1$, $k = 3$, $\alpha = 1$, $\beta = 2$.
24 $p = 1$, $q = 5$, $r = 4$, $m = 2$, $n = 3$, $k = 3$, $\alpha = 2$, $\beta = 1$.
25 $p = 4$, $q = 1$, $r = 3$, $m = 2$, $n = 1$, $k = 5$, $\alpha = 1$, $\beta = 2$.
26 $p = 1$, $q = 3$, $r = 3$, $m = 2$, $n = 2$, $k = 3$, $\alpha = 2$, $\beta = 1$.
27 $p = 3$, $q = 2$, $r = 1$, $m = 2$, $n = 1$, $k = 3$, $\alpha = 1$, $\beta = 2$.
28 $p = 3$, $q = 1$, $r = 3$, $m = 2$, $n = 1$, $k = 3$, $\alpha = 2$, $\beta = 1$.
29 $p = 1$, $q = 3$, $r = 1$, $m = 2$, $n = 2$, $k = 3$, $\alpha = 1$, $\beta = 2$.
30 $p = 4$, $q = 3$, $r = 1$, $m = 2$, $n = 3$, $k = 3$, $\alpha = 2$, $\beta = 1$.

# 3 Лабораторная работа № 3. Дифференцирование в Mathcad

## *3.1 Теоретические сведения*

Оператор дифференцирования в Mathcad может быть вызван двумя способами: используя функцию «штрих», комбинацией клавиш Ctrl + Alt + ‘, и вызовом функции дифференцирования на панели операторов. Для получения кратной производной необходимо применить нужное количество функций «штрих» либо указать в операторе дифференцирования порядок производной.

### *Контрольные вопросы*

1 Каким образом в Mathcad вызывается оператор дифференцирования?

## *3.2 Задания для самостоятельного выполнения*

1 Имеется дифференциальное уравнение, начальные условия и некоторая функция. Запрограммируйте функцию в Mathcad, которая возвращает 1, если заданная функция является решением задачи Коши для дифференциального уравнения, и 0, если не является.

2 Для заданной функции вычислите первую и вторую производные. Постройте графики всех функций.

3 Напишите программу в Mathcad, которая для заданной функции ищет приближенное разложение в ряд Тейлора в виде суммы первых пяти слагаемых ряда и возвращает его в явном виде.

4 Напишите программу в Mathcad, которая находит производную параметрически заданной функции.

5 Напишите программу в Mathcad, которая находит производную неявно заданной функции.

# 4 Лабораторная работа № 4. Интегрирование в Mathcad

## *4.1 Теоретические сведения*

Для нахождения первообразной или неопределенного интеграла используется оператор интегрирования с панели операторов. В случае, если необходимо вычислить определенный интеграл, следует заполнить числовыми значениями или переменными метки нижнего и верхнего предела.

### *Контрольные вопросы*

1 Каким образом в Mathcad можно выполнить нахождение неопределенного интеграла?

2 Каким образом в Mathcad можно выполнить нахождение определенного интеграла?

### *4.2 Задания для самостоятельного выполнения*

1 Напишите программу в Mathcad, которая реализует функцию вычисления определенного интеграла, принимая на вход подынтегральную функцию и пределы интегрирования.

2 Напишите программу в Mathcad, которая определяет функцию интеграла с переменным верхним пределом и выводит на одном графике функцию и ее интеграл с переменным верхним пределом.

3 Вычислите определенный интеграл $\int_0^2 \frac{\left(4\sqrt{2-x}-\sqrt{x+2}\right)}{\left(\sqrt{x+2}+4\sqrt{2-x}\right)\left(x+2\right)^2}dx$.

4 Вычислите определенный интеграл $\int_0^{\frac{3}{2}} \frac{x^2 dx}{\sqrt{9-x^2}}$.

5 Найдите неопределенный интеграл $\int \frac{\sqrt[5]{1+\sqrt[3]{x}}}{x\sqrt[5]{x^2}}dx$, учитывая, что Mathcad не может напрямую найти аналитическое выражение результата интегрирования.

## 5 Лабораторная работа № 5. Нелинейные алгебраические уравнения в Mathcad

### *5.1 Теоретические сведения*

Для решения нелинейных алгебраических уравнений в Mathcad можно использовать функцию polyroots, которая на вход принимает вектор коэффициентов полинома, определяющего левую часть уравнения. Функция возвращает вектор корней, если их несколько.

В более общем случае для решения трансцендентных уравнений следует использовать функцию root, которая принимает в качестве параметров функцию, определяющую левую часть уравнения, начальное приближение или границы интервала отделения корней. Возвращает только один корень, получить все возможные корни уравнения возможно, изменив начальное приближение или интервал отделения корней.

Для решения уравнений также можно использовать символьные преобразования с ключевым словом solve. Для получения всех корней уравнения необходимо добавить модификатор fully.

### *Контрольные вопросы*

1 Перечислите функции и их параметры, реализованные в Mathcad для решения нелинейных, в том числе и алгебраических, уравнений.

### *5.2 Задания для самостоятельного выполнения*

Напишите программу-функцию в Mathcad, которая для заданного уравнения определяет его корень с заданной точностью методом половинного деления.

Для заданных уравнений отделите корни с помощью построения графика, найдите все возможные корни, используя встроенные возможности Mathcad и написанную функцию. Вариант задания для выполнения совпадает с номером в журнале.

1 $x^3 - 3x^2 + 9x - 8 = 0;\ x - \sin x = 0{,}25.$

2 $x^3 - 6x - 8 = 0;\ \operatorname{tg}(0{,}58x + 0{,}1) = x^2.$

3 $x^3 - 3x^2 + 6x + 3 = 0;\ \sqrt{x} - \cos(0{,}387x) = 0.$

4 $x^3 - 0{,}1x^2 + 0{,}4x - 1{,}5 = 0;\ \operatorname{tg}(0{,}4x + 0{,}4) = x^2.$

5 $x^3 - 3x^2 + 9x + 2 = 0;\ \lg x - \dfrac{7}{(2x+6)} = 0.$

6 $x^3 + x - 5 = 0;\ \operatorname{tg}(0{,}5x + 0{,}2) = x^2.$

7 $x^3 + 0{,}2x^2 + 0{,}5x - 2 = 0;\ 3x - \cos x - 1 = 0.$

8 $x^3 + 3x + 1 = 0;\ x + \lg x = 0{,}5.$

9 $x^3 + 0{,}2x^2 + 0{,}5x - 2 = 0;\ \operatorname{tg}(0{,}5x + 0{,}1) = x^2.$

10 $x^3 - 3x^2 + 12x - 9 = 0;\ x^2 + 4\sin x = 0.$

11 $x^3 - 3x^2 + 12x - 9 = 0;\ \operatorname{ctg} 1{,}05x - x^2 = 0.$

12 $x^3 - 3x^2 + 6x - 2 = 0;\ \operatorname{tg}(0{,}4x + 0{,}3) = x^2.$

13 $x^3 - 0{,}1x^2 + 6x - 1 = 0;\ x\lg x - 1{,}2 = 0.$

14 $x^3 + 3x^2 + 6x - 1 = 0;\ 1{,}8x^2 - \sin 10x - 0{,}1 = 0.$

15 $x^3 + 0{,}1x^2 + 0{,}4x - 1{,}2 = 0;\ \operatorname{ctg} x - \dfrac{x}{4} = 0.$

16 $x^3 + 4x - 6 = 0;\ \operatorname{tg}(0{,}3x + 0{,}4) = x^2.$

17 $x^3 + 0{,}2x^2 + 0{,}5x + 0{,}8 = 0;\ x^2 - 20\sin x + 1 = 0.$

18 $x^3 - 3x^2 + 12x - 12 = 0;\ \operatorname{ctg} x - \dfrac{x}{3} = 0.$

19 $x^3 - 0{,}2x^2 + 0{,}3x + 1{,}2 = 0;\ \operatorname{tg}(0{,}47x + 0{,}2) = x^2.$

20 $x^3 - 2x + 5 = 0;\ x^2 + 4\sin x = 0.$

21 $x^3 - 0,2x^2 + 0,5x - 1,4 = 0$; $\operatorname{ctg} x - \frac{x}{2} = 0$.

22 $x^3 - 3x^2 + 6x - 5 = 0$; $2x - \lg x - 7 = 0$.

23 $x^3 - 0,1x^2 + 0,5x - 1 = 0$; $\operatorname{tg}(0,44x + 0,3) = x^2$.

24 $x^3 - 0,2x^2 + 0,5x - 1 = 0$; $3x - \cos x - 1 = 0$.

25 $x^3 + 3x^2 + 12x + 8 = 0$; $\operatorname{ctg} x - \frac{x}{10} = 0$.

26 $x^3 - 0,1x^2 + 0,4x + 2 = 0$; $x^2 + 4\sin x + 0,1 = 0$.

27 $x^3 - 0,2x^2 + 0,4x - 1,4 = 0$; $\operatorname{tg}(0,36x + 0,4) = x^2$.

28 $x^3 + 0,4x^2 + 0,6x - 1,6 = 0$; $x + \lg x = 0,5$.

29 $x^3 + x - 3 = 0$; $\operatorname{ctg} - \frac{x}{5} = 0$.

30 $x^3 - 0,2x^2 + 0,5x + 1,4 = 0$; $\operatorname{tg} x - \frac{x}{2} + 1 = 0$.

# 6 Лабораторная работа № 6. Оптимизация в Mathcad

## *6.1 Теоретические сведения*

Для решения задач на отыскание оптимальных (наибольших или наименьших) значений функций одной или нескольких переменных в Mathcad имеются функции maximize и minimize, принимающие в качестве параметров функцию (в операторной форме) и независимые переменные. Функции могут выполнять в блоке решения или без него, для выполнения поиска оптимальных значений требуется инициализация переменных. При использовании в блоке решения на независимые переменные могут налагаться дополнительные ограничения.

### *Контрольные вопросы*

1 Опишите оптимизационные функции, встроенные в Mathcad.

## *6.2 Задания для самостоятельного выполнения*

Решите задачу минимизации функции $f(x,y) = (x-a)^2 + b \cdot e^{(y-a)} + d \cdot x \cdot y$ с ограничением в виде равенства $d \cdot x + c \cdot y^2 = f$. Решите задачу методом множителей Лагранжа и с помощью встроенных методов оптимизации Mathcad, сравните результаты. Номер варианта совпадает с номером в журнале.

1 $a = 3$, $b = 2$, $c = 1$, $d = 2$, $f = 9$.

2 $a = 2$, $b = 3$, $c = 2$, $d = 1$, $f = 9$.

3 $a = 2, b = 1, c = 1, d = 1, f = 4$.
4 $a = 2, b = 1, c = 1, d = 2, f = 2$.
5 $a = 1, b = 4, c = 4, d = 4, f = 2$.
6 $a = 1, b = 4, c = 2, d = 4, f = 3$.
7 $a = 1, b = 4, c = 2, d = 8, f = 6$.
8 $a = 2, b = 2, c = 2, d = 5, f = 10$.
9 $a = 2, b = 1, c = 1, d = 2, f = 10$.
10 $a = 1, b = 2, c = 1, d = 2, f = 2$.
11 $a = 3, b = 2, c = 1, d = 2, f = 9$.
12 $a = 2, b = 3, c = 2, d = 1, f = 9$.
13 $a = 2, b = 1, c = 1, d = 1, f = 4$.
14 $a = 2, b = 1, c = 1, d = 2, f = 2$.
15 $a = 1, b = 4, c = 4, d = 4, f = 2$.
16 $a = 1, b = 4, c = 2, d = 4, f = 3$.
17 $a = 1, b = 4, c = 2, d = 8, f = 6$.
18 $a = 2, b = 2, c = 2, d = 5, f = 10$.
19 $a = 2, b = 1, c = 1, d = 2, f = 10$.
20 $a = 1, b = 2, c = 1, d = 2, f = 2$.
21 $a = 3, b = 2, c = 1, d = 2, f = 9$.
22 $a = 2, b = 3, c = 2, d = 1, f = 9$.
23 $a = 2, b = 1, c = 1, d = 1, f = 4$.
24 $a = 2, b = 1, c = 1, d = 2, f = 2$.
25 $a = 1, b = 4, c = 4, d = 4, f = 2$.
26 $a = 1, b = 4, c = 2, d = 4, f = 3$.
27 $a = 1, b = 4, c = 2, d = 8, f = 6$.
28 $a = 2, b = 2, c = 2, d = 5, f = 10$.
29 $a = 2, b = 1, c = 1, d = 2, f = 10$.
30 $a = 1, b = 2, c = 1, d = 2, f = 2$.

# 7 Лабораторная работа № 7. Линейная алгебра в Mathcad

## *7.1 Теоретические сведения*

Mathcad располагает широким набором функций для работы с матрицами и векторами, все они доступны в панели «Матрицы/таблицы» основной ленты. В общем случае матрицы и векторы в Mathcad представлены в виде массивов. С помощью функций можно создавать и разделять массивы, находить характеристики массива, выполнять разложение матрицы, а также множество других действий.

Основные функции для работы с векторами и матрицами:

– **cols**, **rows** – рассчитывают размерность матрицы;

– **last**, **length** – рассчитывают размерность вектора;

– **max**, **min** – возвращают наибольшее и наименьшее значение в матрице;

– **cond1**, **cond2**, **conde**, **condi** – возвращают число обусловленности матрицы в соответствии с различными нормами;

– **norm1**, **norm2**, **norme**, **normi** – возвращают норму матрицы;

– **geninv**, **rank**, **rref** – рассчитывают ранг матрицы, левую обратную или приведенно-ступенчатую форму по строкам;

– **eigenvals**, **eigenvec**, **eigenvecs**, **genvals**, **genvecs**, **tr** – рассчитывают собственные значения, собственные векторы или след квадратной матрицы;

– **Cholesky** – вычисляет Cholesky–квадратный корень из матрицы;

– **LU** – вычисляет LU-разложение на нижнюю и верхнюю треугольные матрицы;

– **QR** – вычисляет QR-разложение на ортонормированную и верхнюю треугольные матрицы;

– **svd** – вычисляет разложение матрицы по сингулярным значениям.

С остальными функциями можно ознакомиться во вкладке «Матрицы/таблицы», используя закладку «Функции с векторами/матрицами».

### *Контрольные вопросы*

1 С помощью какой команды можно сформировать матрицу, каждый элемент которой является функцией от его индексов?

2 С помощью какой команды можно извлечь подматрицу из матрицы?

3 С помощью какой команды можно построить матрицу, содержащую на главной диагонали элементы некоторого вектора?

### *7.2 Задания для самостоятельного выполнения*

1 Создайте функцию **rmatr**(**n**), возвращающую матрицу размерности 2*2, содержащую случайные числа от 0 до n в качестве элементов (использовать функцию **rnd**()). Решите матричное уравнение $3\cdot(\mathrm{A}+\mathrm{B})\cdot\mathrm{C}-2\mathrm{X}^{\mathrm{T}}=\frac{1}{7}|\mathrm{D}|\cdot\mathrm{F}$, где матрицы A, B, C, D, F – случайные матрицы размерности 2*2; X – искомая матрица.

2 Выполните следующие действия:

– создайте матрицу A заданной размерности $m$*$n$ ($m < n$);

– транспонируйте матрицу A;

– получите из матрицы $i$-й и $j$-й столбцы, найдите их скалярное произведение;

– примените к каждому элементу матрицы A функцию $z(x)$;

– создайте матрицу B размерностью $n$*$m$, у которой элементы являются функциями своих индексов;

– вычислите произведение A*B;

– из матрицы A получите квадратную матрицу C размерностью $(m-1)*(m-1)$, для которой найдите собственные числа и собственные векторы.

3 Выполните приведение квадратической формы $F(x,y,z) = ax^2 + by^2 + cz^2 + dxy + exz + fyz$ к канонической форме с помощью ортогональной матрицы. Выполните проверку. Номер варианта совпадает с номером в журнале, задания представлены в таблице 7.1.

Таблица 7.1 – Варианты заданий

| Вариант | *a* | *b* | *c* | *d* | *e* | *f* |
|---|---|---|---|---|---|---|
| 1 | 0,5 | 0 | 0,5 | $2\sqrt{2}$ | 0 | $-2\sqrt{2}$ |
| 2 | 1,5 | 2 | 1,5 | 0 | –1 | 0 |
| 3 | 1,5 | 2 | 1,5 | $\sqrt{2}$ | –1 | $-\sqrt{2}$ |
| 4 | 1,5 | 2 | 1,5 | $-\sqrt{2}$ | 1 | $-\sqrt{2}$ |
| 5 | 0 | 2 | 0 | $-\sqrt{2}$ | 4 | $-\sqrt{2}$ |
| 6 | –0,5 | 1 | –0,5 | $-2\sqrt{2}$ | 3 | $-2\sqrt{2}$ |
| 7 | –0,5 | 1 | –0,5 | $2\sqrt{2}$ | –3 | $-2\sqrt{2}$ |
| 8 | 0 | 2 | 0 | $2\sqrt{2}$ | –4 | $-2\sqrt{2}$ |
| 9 | 0 | 2 | 0 | $-2\sqrt{2}$ | 4 | $-2\sqrt{2}$ |
| 10 | 2 | 2 | 2 | –2 | 0 | $-2\sqrt{2}$ |
| 11 | 0,25 | 0,5 | 0,25 | $-3,5\sqrt{2}$ | 0,5 | $-3,5\sqrt{2}$ |
| 12 | 0,75 | 1,5 | 0,75 | $-2,5\sqrt{2}$ | 1,5 | $-2,5\sqrt{2}$ |
| 13 | 0,5 | 0 | 0,5 | $2\sqrt{2}$ | 0 | $-2\sqrt{2}$ |
| 14 | 1,5 | 2 | 1,5 | 0 | –1 | 0 |
| 15 | 1,5 | 2 | 1,5 | $\sqrt{2}$ | –1 | $-\sqrt{2}$ |
| 16 | 1,5 | 2 | 1,5 | $-\sqrt{2}$ | 1 | $-\sqrt{2}$ |
| 17 | 0 | 2 | 0 | $-\sqrt{2}$ | 4 | $-\sqrt{2}$ |
| 18 | –0,5 | 1 | –0,5 | $-2\sqrt{2}$ | 3 | $-2\sqrt{2}$ |
| 19 | –0,5 | 1 | –0,5 | $2\sqrt{2}$ | –3 | $-2\sqrt{2}$ |
| 20 | 0 | 2 | 0 | $2\sqrt{2}$ | –4 | $-2\sqrt{2}$ |
| 21 | 0 | 2 | 0 | $-2\sqrt{2}$ | 4 | $-2\sqrt{2}$ |
| 22 | 2 | 2 | 2 | –2 | 0 | $-2\sqrt{2}$ |
| 23 | 0,25 | 0,5 | 0,25 | $-3,5\sqrt{2}$ | 0,5 | $-3,5\sqrt{2}$ |
| 24 | 0,75 | 1,5 | 0,75 | $-2,5\sqrt{2}$ | 1,5 | $-2,5\sqrt{2}$ |
| 25 | 0,5 | 0 | 0,5 | $2\sqrt{2}$ | 0 | $-2\sqrt{2}$ |
| 26 | 1,5 | 2 | 1,5 | 0 | –1 | 0 |
| 27 | 1,5 | 2 | 1,5 | $\sqrt{2}$ | –1 | $-\sqrt{2}$ |
| 28 | 1,5 | 2 | 1,5 | $-\sqrt{2}$ | 1 | $-\sqrt{2}$ |
| 29 | 0 | 2 | 0 | $-\sqrt{2}$ | 4 | $-\sqrt{2}$ |
| 30 | –0,5 | 1 | –0,5 | $-2\sqrt{2}$ | 3 | $-2\sqrt{2}$ |

# 8 Лабораторная работа № 8. Системы линейных уравнений в Mathcad

## *8.1 Теоретические сведения*

Для решения систем линейных уравнений в Mathcad можно использовать блок решения, описав в разделе «ограничения» систему линейных уравнений. Альтернативным способом решения систем линейных уравнений является функция lsolve. Также для нахождения решений системы линейных алгебраических уравнений в символьной форме используется оператор символьных преобразований с ключевым словом solve и указанием переменных, по которым необходимо решить систему. Уравнения системы должны быть записаны без правой части (линейные многочлены от неизвестных) как элементы некоторого вектора. В случае, если система имеет бесконечное число решений, следует добавить модификатор fully.

### *Контрольные вопросы*

1 Опишите, каким образом в Mathcad можно получить все возможные решения системы линейных алгебраических уравнений в случае, если ранг расширенной матрицы системы меньше числа неизвестных, входящих в систему.

2 Какие действия необходимо выполнить для того, чтобы решить систему линейных алгебраических уравнений с помощью блока решения?

## *8.2 Задания для самостоятельного выполнения*

Решите систему линейных алгебраических уравнений с параметром

$$\begin{cases} a \cdot x + (k-15) \cdot y + 2 \cdot z = 2, \\ k \cdot x + (a-3) \cdot y + 3 \cdot z = k - 10, \\ -k \cdot x + 4 \cdot y + (a-2) \cdot z = 10 - k. \end{cases}$$

Параметр $k$ совпадает с номером по журналу.

# 9 Лабораторная работа № 9. Основы программирования в Octave. Работа с файлами. Графическая система

## *9.1 Теоретические сведения*

В GNU Octave встроен язык программирования, поддерживающий основные принципы программирования. Программы сохраняются в файлах с расширением .m (.M) и выполняются Octave запуском из командной строки автоматически.

Оператор присваивания служит для определения новой переменной. Чтобы определить новую переменную, достаточно присвоить ей значение через знак равенства. Любую переменную Octave воспринимает как матрицу. В простейшем случае матрица может содержать одну строку и один столбец (число).

Ввод исходных данных в m-программу можно произвести командой

**var_name = input(‘prompt’)**,

где **var_name** – имя переменной; **‘prompt’** – строка-подсказка.

Для вывода результатов используется функция

**disp(‘string’)**

**disp(var_name)**.

Условный оператор в Octave имеет следующий синтаксис:

```
if cond
   commands1
else
   commands2
end
```

Здесь **cond** – логическое выражение; **commands1**, **commands2** – операторы языка или встроенные функции Octave. Условный оператор может быть расширен одной или несколькими ветвями **elseif**.

Оператор альтернативного выбора имеет синтаксис

```
switch param
case value1
   commands1
case value2
   commands2
…
otherwise
   commands
end
```

Оператор **switch** в зависимости от значения **param** выбирает одну из имеющихся ветвей.

Условный циклический оператор имеет синтаксис

```
while expr
   commands
end
```

и выполняется до тех пор, пока выражение **expr** истинно.

Оператор цикла с известным числом повторений

**for param = begin_value: step: end_value**
**commands**
**end**

Оператор цикла выполняет фиксированное количество повторений, присваивая переменной **param** значения от **begin_value** до **end_value** с шагом **step**.

Операторы передачи управления **break** и **continue** используются только внутри циклов **while**, **for** и управление передается оператору, следующему непосредственно за циклом. Оператор continue начинает новую итерацию цикла, даже если предыдущая не была завершена.

Ввод массивов следует организовывать поэлементно.

В Octave присутствует множество функций для работы со строками:

– **char(code)** – возвращает символ по его коду;

– **deblank(s)** – формирует новую пустую строку путем добавления пробелов в конце строки s;

– **int2str(x)** – преобразует число к целому типу и запись результатов в строковый массив;

– **findstr(str, substr)** – возвращает номер позиции, начиная с которой подстрока **substr** входит в строку **str**;

– **lower(s)** – возвращает строку путем преобразования s к строчным буквам;

– **mat2str(x,n)** – преобразует числовую матрицу x в строку, каждый элемент матрицы округляется до n значащих цифр после запятой;

– **sprint(format, x)** – формирует строку из чисел, хранящихся в переменной x, в соответствии с форматом **format**;

– **sscanf(s, format)** – функция возвращает из строки s числовое значение или массив значений в соответствии с форматом;

– **str2double(s)** – формирует число из строки s, если это возможно;

– **strcat(s1, s2, …, sn)** – формируется строка сцепкой строк s1, s2, …;

– **strcmp(s1, s2)** – сравнивает строки s1 и s2, не различая строчные и прописные буквы;

– **upper(s)** – возвращает строку s, преобразованную к прописным буквам.

Открытие текстового файла осуществляется командой

**fopen(filename, mode)**.

Здесь **filename** – строка, в которой хранится имя файла с указанием пути к нему; **mode** – режим работы с файлом, может принимать значения ‘rt’, ‘rt+’, ‘wt’, ‘wt+’, ‘at’, ‘at+’, для открытия файла в режиме чтения, записи, добавления информации. Функция **fopen** возвращает идентификатор файла.

Для форматированного вывода информации в файл можно использовать функцию

**fprintf(f, s1, s2)**.

Здесь f – идентификатор файла; s1 – строка вывода (формат); s2 – список выводимых переменных.

Для считывания информации из файла служит функция

**fscanf(f, s1, n)**,

для которой f – идентификатор файла; s1 – строка форматов; n – количество считываемых значений.

По окончании операций с файлом обязательно выполнение команды **fclose(f)** во избежание потери данных.

Организовать чтение числовых данных из файла в матрицу можно с помощью функции **dlmread**. Вывод числовых данных из матрицы в файл осуществляется с помощью функции **dlmwrite**.

Вывод графиков в Octave осуществляется с помощью функции **plot**:

**plot(x, y, property)**.

Здесь x – вектор абсцисс; y – вектор ординат; property – строка, описывающая свойства графика.

На одном изображении можно вывести несколько графиков, для этого необходимо указать последовательно пары векторов абсцисс и ординат.

Для того чтобы построить изображение поверхности, определенной уравнением $z = f(x, y)$, необходимо построить векторы значений независимых переменных, с помощью функции **[X, Y] = meshgrid(x, y)** построить сетку значений, вычислить значения функции $z = f(X, Y)$, построить график с помощью функции **surf(x, y, z)**.

Для объявления пользовательской файл-функции следует использовать следующие команды:

```
function [ret-list] = name(arg-list)
   commands
endfunction
```

Здесь ret-list – перечень переменных с возвращаемыми функцией значениями; arg-list – перечень передаваемых в функцию переменных; commands – последовательность выполняемых команд.

Удобным способом является объявление анонимных функций:

**name = @(var [, var2,…]) function**.

Доступ к такой функции осуществляется по ее имени, также можно использовать переменные Octave в качестве ее значений. Такое объявление можно использовать внутри других функций.

Для отображения графика анонимно объявленной функции используется функция **fplot**.

### *Контрольные вопросы*

1 Опишите синтаксис условного оператора в Octave.

2 Опишите синтаксис оператора цикла с фиксированным числом повторений в Octave.

3 Опишите синтаксис оператора цикла с предусловием в Octave.

4 Каким образом в Octave осуществляется считывание информации из файла?

5 Какие функции в Octave используются для вывода графиков?

### *9.2 Задания для самостоятельного выполнения*

1 Создайте m-скрипт в Octave, реализующий алгоритм половинного деления решения нелинейного уравнения.

2 Создайте m-скрипт в Octave, реализующий алгоритм метода хорд и касательных (комбинированный метод) решения нелинейного уравнения.

3 Создайте m-скрипт в Octave, реализующий алгоритм метода итераций решения нелинейного уравнения.

4 С помощью созданных скриптов решите уравнения из лабораторной работы № 5 в соответствии с номером варианта.

## 10 Лабораторная работа № 10. Линейная алгебра и аналитическая геометрия в Octave

### *10.1 Теоретические сведения*

Векторы и матрицы в Octave задаются путем ввода их элементов. Элементы строк разделяются пробелами или запятыми, элементы столбцов разделяются символом «точка с запятой», при этом конструкцию заключают в квадратные скобки. В качестве элементов вектора или матрицы могут выступать другие векторы или матрицы.

Обратиться к элементам вектора (матрицы) можно по индексам, указав их в круглых скобках после имени вектора через запятую (матрицы). Получить несколько строк или столбцов одновременно можно, используя символ двоеточие, или так называемый срез матрицы. Упоминание в качестве индекса элемента матрицы двоеточия позволяет вывести сразу весь столбец или строку матрицы целиком.

Сложение векторов или матриц осуществляется поэлементно, доступно только для операндов согласованных размерностей. Сложение двух матриц осуществляется с помощью знака «+».

Для транспонирования матрицы используется знак апострофа.

Умножение матрицы на число или матрицы на матрицу осуществляется с помощью знака «*». В случае перемножения матриц они должны быть согласованными.

Деление матрицы на число осуществляется с помощью знака «/».

Octave поддерживает векторизованные вычисления, т. е. применение функций или других операций сразу ко всем элементам вектора или матрицы.

Поэлементное умножение векторов или матриц осуществляется с помощью оператора «.*».

Поэлементное деление векторов или матриц осуществляется с помощью оператора «./».

Поэлементное возведение вектора или матрицы в степень осуществляется с помощью оператора «.^».

Операция A/B в случае двух матричных операндов эквивалентна операции $A \cdot B^{-1}$. Операция A\B эквивалентна операции $A^{-1} \cdot B$.

Функции для работы с векторами:

– **length(x)** – определяет длину вектора x;

– **prod(x)** – вычисляет произведение элементов вектора x;

– **cumprod(x)** – формирует вектор той же размерности, что и x, каждый элемент которого является кумулятивным произведением предыдущих элементов исходного вектора;

– **sum(x)** – вычисляет сумму элементов вектора x;

– **cumsum(x)** – формирует вектор с кумулятивной суммой элементов вектора x;

– **diff(x)** – формирует вектор размерностью на единицу меньше, чем x, каждый элемент которого равен разности двух подряд идущих элементов вектора x;

– **min(x)** – вычисляет наименьший элемент вектора x;

– **max(x)** – вычисляет наибольший элемент вектора x;

– **mean(x)** – вычисляет арифметическое среднее элементов x;

– **dot(x1, x2)** – вычисляет скалярное произведение векторов x1, x2;

– **cross(x1, x2)** – вычисляет векторное произведение векторов x1, x2;

– **sort(x)** – сортирует элементы вектора x; для сортировки по убыванию необходимо сортировать вектор –x.

Функции для работы с матрицами:

– **eye(n [, m])** – возвращает единичную (квазиединичную) матрицу размерности n*n (n*m);

– **ones(n [, n, p, …])** – возвращает матрицу требуемой размерности, состоящую из единиц;

– **zeros(n [, m, p, …])** – возвращает нулевую матрицу требуемой размерности;

– **diag(x [, k])** – возвращает квадратную матрицу с элементами x на главной или k-й диагонали, x – вектор или матрица;

– **rand([n, m, p, …])** – возвращает матрицу требуемой размерности с элементами, распределенными по равномерному закону из интервала (0, 1), без аргументов возвращает случайное число;

– **randn([n, m, p, …])** – возвращает матрицу требуемой размерности, элементы которой являются случайными числами, распределенными нормально с параметрами (0, 1);

– **linspace(a, b [, n])** – возвращает массив из 100 или из n чисел, равномерно распределенных между числами a и b;

– **logspace(a, b [, n])** – формирует массив из 50 или n точек, равномерно распределенных в логарифмическом масштабе между числами 10a и 10b;

– **repmat(M, n [, m])** – формирует матрицу, состоящую из n*n (n*m) копий матрицы (скаляра) M;

– **reshape(M, m, n)** – возвращает матрицу размерности m*n, сформированную из матрицы M путем последовательной выборки по столбцам;

– **cat(n, A, B [, C, …])** – возвращает матрицу, полученную объединением матриц A, B, …, по столбцу (n = 1) или строке (n = 2);

– **rot90(M [, k])** – выполняет поворот матрицы M на 90 градусов (k раз);

– **tril(M [, k])** – формирует из матрицы M нижнюю треугольную матрицу, начиная с главной или k-й диагонали;

– **triu(M [, k])** – формирует из матрицы M верхнюю треугольную матрицу, начиная с главной или k-й диагонали;

– **size(M)** – определяет число строк и столбцов матрицы M, результатом ее работы является вектор;

– **prod(M [, k])** – формирует вектор-строку или вектор-столбец, в зависимости от значения k, каждый элемент которой является произведением соответствующего столбца или строки матрицы M;

– **cumprod(M [, k])** – формирует вектор-строку или вектор-столбец кумулятивных произведений;

– **sum(M [, k])** – формирует вектор-строку или вектор-столбец, каждый элемент которого является суммой элементов соответствующего столбца или строки матрицы M;

– **cumsum(M [, k])** – формирует вектор-строку или вектор-столбец кумулятивных сумм;

– **diff(M)** – формирует матрицу, содержащую на одну строку меньше, чем исходная, содержащую разности элементов соседних строк исходной матрицы;

– **min(M)** – формирует вектор-строку, каждый элемент которой есть наименьший элемент соответствующего столбца, может возвращать вектор, первое значение которого – вектор наименьших значений, второе – вектор номеров строк, в которых эти элементы находятся;

– **min(M, [], k)** – формирует вектор с наименьшими значениями, параметр k позволяет управлять направлением поиска;

– **min(M)** – формирует вектор-строку, каждый элемент которой есть наибольший элемент соответствующего столбца, может возвращать вектор, первое значение которого вектор наибольших значений, второе – вектор номеров строк, в которых эти элементы находятся;

– **min(M, [], k)** – формирует вектор с наибольшими значениями, параметр k позволяет управлять направлением поиска;

– **max(M)** – формирует вектор-строку, каждый элемент которой есть наибольший элемент соответствующего столбца, может возвращать вектор, первое значение которого – вектор наибольших значений, второй – вектор номеров строк, в которых эти элементы находятся;

– **max(M, [], k)** – формирует вектор с наибольшими значениями, параметр k позволяет управлять направлением поиска;

– **mean(M [, k])** – формирует вектор-строку или вектор-столбец, в зависимости от значения k, каждый элемент которого является средним значением элементов соответствующего столбца или строки матрицы M;

– **sort(M)** – возвращает матрицу той же размерности, что и M, каждый столбец которой упорядочен по возрастанию;

– **sqrtm(M)** – возвращает матрицу X, для которой X*X = A;

– **expm(M)** и **logm(M)** – вычисляют матричную экспоненту и матричный логарифм.

Функции, реализующие численные алгоритмы решения задач линейной алгебры:

– **det(M)** – вычисляет определитель квадратной матрицы M;

– **trace(M)** – вычисляет след матрицы M;

– **norm(M [, p])** – возвращает различные нормы матрицы M в зависимости от p;

– **cond(M [, p])** – возвращает число обусловленности матрицы M, основанное на норме p;

– **rcond(M)** – вычисляет величину, обратную **cond(M)**;

– **inv(M)** – возвращает матрицу, обратную к M;

– **eig(M)** – возвращает вектор собственных значений матрицы M в формате **[Matr, D]**, где Matr – матрица, столбцы которой содержат собственные векторы M; D – диагональная матрица, содержащая собственные значения M;

– **poly(M)** – возвращает вектор-строку характеристического полинома матрицы M;

– **rref(M)** – осуществляет приведение матрицы M к треугольной форме, используя метод исключения Гаусса;

– **chol(M)** – возвращает разложение по Холецкому для положительно определенной симметрической матрицы M;

– **lu(M)** – выполняет LU-разложение, вывод в формате **[L, U, P]**, где L – нижняя треугольная матрица; U – верхняя треугольная; P – матрица перестановок;

– **qr(M)** – выполняет QR-разложение, возвращает результат в виде **[Q, R, P]**, где Q – ортогональная матрица; R – верхняя треугольная матрица; P – матрица перестановок;

– **svd(M)** – возвращает вектор сингулярных чисел матрицы M, результат в формате **[U, S, V]**, где U – ортонормированные собственные векторы, соответствующие наибольшим собственным значениям матрицы M; V – ортонормированные собственные векторы матрицы $M*M^T$; S – диагональная матрица из сингулярных чисел (неотрицательных значений квадратных корней из собственных значений матрицы $M*M^T$).

Ввиду того, что алгебраические векторы представляют собой матрицы-строки или матрицы-столбцы, Octave позволяет реализовать математический аппарат векторной алгебры и аналитической геометрии.

### *Контрольные вопросы*

1 Каким образом в Octave задаются векторы и матрицы?

2 Перечислите функции для работы с векторами и матрицами.

3 Перечислите функции, реализующие вычислительные алгоритмы линейной алгебры.

### *10.2 Задания для самостоятельного выполнения*

1 Используя Octave, выполните задания лабораторной работы № 7.

2 Четыре точки заданы своими координатами в трехмерном пространстве. Создайте m-скрипт, который вычисляет объем пирамиды, построенной, определяемой этими точками, длины высот, а также строит изображение пирамиды и ее высот. Координаты точек задаются равномерно распределенными случайными числами из диапазона [–10; 10]. Для построения изображения используйте функцию plot3.

## 11 Лабораторная работа № 11. Нелинейные уравнения в Octave

### *11.1 Теоретические сведения*

В случае, если требуется решить уравнение, левая часть которого представляет собой полином, следует использовать функцию **roots(p)**, при этом полином задается коэффициентами в виде вектора p, начиная от коэффициента при старшей степени.

Для решения трансцендентных уравнений в Octave служит функция

**fzero(name, x0)**.

Здесь name – имя функции, вычисляющей левую часть уравнения; x0 – начальное приближение к корню или интервал изоляции корня.

С помощью функций пакета символьных вычислений symbolic можно выполнить решение уравнения в символьной форме. Для этого необходимо загрузить пакет symbolic командой **pkg load symbolic**, определить символьную переменную командой **syms x** или **x = sym(“x”)** и с помощью функции **solve** выполнить решение уравнения. Синтаксис команды

**solve(fun, var)**.

Здесь fun – символьное выражение, задающее левую часть уравнения, либо само уравнение с логическим оператором сравнения “==”; var – переменные, относительно которых необходимо решить уравнение fun.

### *Контрольные вопросы*

1 Перечислите методы решения нелинейных уравнений, доступные в Octave.

2 Опишите процесс символьного решения уравнения.

### *11.2 Задания для самостоятельного выполнения*

Средствами Octave выполните задания лабораторной работы № 5.

## 12 Лабораторная работа № 12. Интегрирование и дифференцирование в Octave

### *12.1 Теоретические сведения*

Дифференцирование в Octave осуществляется в технике символьных переменных. В функциях интегрирования реализованы различные численные алгоритмы.

Символьное дифференцирование осуществляется с помощью функции diff пакета symbolic:

**diff(f,x,n)**.

Здесь f – символьная функция; x – переменная дифференцирования; n – порядок производной (опционально).

В случае, если функция f задана таблично, в результате выполнения операции **diff** будет получен вектор значений разностей соседних значений функции, для приближенного вычисления значений производной необходимо выполнить деление полученных значений на шаг изменения аргумента. Известно, что точность полученной производной будет выше, если значение шага изменения независимой переменной достаточно близко к нулю.

Символьное интегрирование осуществляется с помощью функции **int** пакета symbolic:

**int(f,x [, a, b])**.

Здесь f – символьная функция; x – переменная, по которой ведется интегрирование; a, b – пределы интегрирования (для определенного интеграла).

Для загрузки пакета symbolic необходимо выполнить команду

**pkg load symbolic**.

Octave позволяет выполнить численное интегрирование методом трапеций, методом Симпсона и с помощью квадратурных формул Гаусса.

Метод трапеций реализован функциями **trapz**, **cumtrapz**:

**trapz(Y)**;
**trapz(X, Y)**;
**trapz(…, DIM)**;
**cumtrapz(Y)**;
**cumtrapz(X, Y)**;
**cumtrapz(…, DIM)**.

В качестве входных параметров могут выступать: вектор значений функции Y; векторы координат X и значений функции Y; DIM – параметр, определяющий размерность решения. Функция **cumtrapz** возвращает вектор промежуточных расчетных значений по методу трапеций.

Интегрирование по методу Симпсона реализовано с помощью функции **quadv**:

**[F, K] = quadv(name, a, b [, tol, trace])**.

Здесь name – символьное имя интегрируемой функции; a, b – пределы интегрирования; tol – значение точности вычислений; F – значение интеграла; K – количество итераций.

Интегрирование с помощью квадратуры Гаусса реализовано посредством функции **quad**:

**[F, kod, K, err] = quad(name, a, b, tol, sing)**.

Здесь name – имя функции, задающей интегральное выражение; a, b – пределы интегрирования; tol – точность вычислений; sing – вектор значений, вблизи которых функция терпит разрыв; F – значение интеграла; kod – код завершения вычислений; K – количество итераций; err – погрешность вычислений.

Численное интегрирование также можно осуществить с помощью функции **integral**, которая позволяет вычислять интегралы от анонимно заданной функции. Синтаксис функции

**integral(fun, xmin, xmax)**.

Здесь fun – имя функции или ее анонимное объявление; xmin, xmax – пределы интегрирования.

Для вычисления двойного и тройного интегралов от анонимно объявленной функции применяются функции **dblquad**, **triplequad**.

### *Контрольные вопросы*

1 Опишите алгоритм численного дифференцирования и его реализацию средствами Octave.

2 Каким образом можно выполнить символьное дифференцирование в Octave?

3 Перечислите функции для численного интегрирования в Octave.

4 Каким образом выполняется символьное интегрирование в Octave?

## *12.2 Задания для самостоятельного выполнения*

1 Напишите m-скрипт, который выполняет численное дифференцирование параметрически заданной функции.

2 Напишите m-скрипт, который позволяет вычислить производную неявно заданной функции символьными преобразованиями.

3 Напишите m-скрипт, который для заданной таблично функции вычисляет первую и вторую производные и строит графики функции и ее производных.

4 Выполните задания 3–5 лабораторной работы № 4 средствами Octave.

# 13 Лабораторная работа № 13. Решение оптимизационных задач в Octave

## *13.1 Теоретические сведения*

Для решения классических оптимизационных задач с ограничениями в Octave можно воспользоваться функцией

**[x, obj, info, iter] = sqp(x0, phi, f, h, lb, ub, maxiter, tol)**,

которая предназначена для решения следующей оптимизационной задачи: найти минимум функции **phi** при следующих ограничениях: g(x) = 0, h(x) >= 0, lb <= x <= ub. Функция **sqp** при решении оптимизационной задачи использует метод квадратичного программирования.

Аргументы **sqp**:

– **x0** – начальное приближение x;

– **phi** – оптимизируемая функция;

– **g** и **h** – функция ограничений g(x) = 0 и h(x) >= 0;

– **lb** и **ub** – нижняя и верхняя границы ограничения;

– **maxiter** – максимальное количество итераций, используемое при решении задачи, по умолчанию равно 100;

– **tol** – точность, при достижении которой вычисления завершаются;

– **x** – точка, в которой функция достигает своего минимального значения;

– **obj** – минимальное значение функции;

– **info** – параметр, характеризующий корректность решения задачи;

– **iter** – реальное количество итераций при решении задачи.

### *Контрольные вопросы*

1 Опишите синтаксис и перечислите параметры функции **sqp**.

## *13.2 Задания для самостоятельного выполнения*

Решите средствами Octave задания лабораторной работы № 6.

# 14 Лабораторная работа № 14. Интерполяция функций в Octave

## *14.1 Теоретические сведения*

Для интерполяции функций можно реализовать известные методы средствами программирования на m-языке либо воспользоваться встроенными возможностями Octave – функциями **interp1** и **interp2**. Функция **interp1** реализует одномерную интерполяцию, функция **interp2** – двумерную.

Синтаксис функции **interp1**

**YI = interp1(X, Y, XI)**;

**YI = interp1(Y, XI)**;

**YI = interp1(…, METHOD)**;

**YI = interp1(…, EXTRAP)**.

Функция возвращает значения YI приближенной функции в точках XI по начальным данным X, Y. В случае, если отсутствуют значения аргумента X, значения Y индексируются по их номерам в векторе.

В качестве параметра METHOD могут быть использованы следующие значения:

– **nearest** – показывает ближайшие соседние точки;

– **previous** – показывает предыдущую соседнюю точку;

– **next** – показывает следующую соседнюю точку;

– **linear** – строит линейную интерполяцию по соседним точкам;

– **pchip** – строит кубическую интерполяцию Эрмита по полиномам с гладкой первой производной;

– **cubic** – строит кубическую интерполяцию;

– **spline** – интерполирует сплайнами с гладкой первой и второй производными.

В случае, если необходимо получить значения интерполяционной функции за пределами интервала, определенного значениями X, параметр EXTRAP необходимо определить как строку **extrap**.

Интерполяция функции двух переменных осуществляется с помощью функции **interp2**. С особенностями ее применения можно ознакомиться в справочной системе Octave.

### *Контрольные вопросы*

1 Что такое интерполирование?

2 Каким образом можно интерполировать функцию в Octave?

3 Какая встроенная функция имеется в Octave для интерполирования функций?

## *14.2 Задания для самостоятельного выполнения*

В результате эксперимента получены значения некоторой функции $y(x)$ в зависимости от значений параметра $x$. Постройте график интерполяционной зависимости, используя линейный и кубический сплайн, вычислите ожидаемые значения в заданных точках $x^*$. Номер варианта соответствует номеру студента в журнале, варианты заданий представлены в таблице 14.1.

Таблица 14.1 – Варианты заданий

| Номер варианта | Исходные данные | | | | | | | $x^*$ |
|---|---|---|---|---|---|---|---|---|
| 1 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,4 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 2 | $x_i$ | 1,51 | 1,55 | 1,57 | 1,59 | 1,64 | 1,72 | 1,8 |
| | $y_i$ | 3,281 | 3,521 | 3,634 | 3,782 | 3,801 | 3,921 | |
| 3 | $x_i$ | 4,51 | 4,722 | 4,823 | 4,919 | 4,963 | 5,023 | 5,25 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 4 | $x_i$ | 5,667 | 5,802 | 5,836 | 5,921 | 6,308 | 6,322 | 6,5 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 5 | $x_i$ | 0,2928 | 0,3103 | 0,3313 | 0,3476 | 0,3623 | 0,383 | 0,4 |
| | $y_i$ | 3,247 | 3,174 | 3,118 | 3,042 | 2,985 | 2,915 | |
| 6 | $x_i$ | 5,4667 | 5,6802 | 5,5836 | 5,921 | 6,1308 | 6,2322 | 6,3 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 7 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,39 |
| | $y_i$ | 3,281 | 3,521 | 3,634 | 3,782 | 3,801 | 3,921 | |
| 8 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,34 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 9 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,4 |
| | $y_i$ | 3,247 | 3,174 | 3,118 | 3,042 | 2,985 | 2,915 | |
| 10 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,37 |
| | $y_i$ | 5,4667 | 5,6802 | 5,5836 | 5,921 | 6,1308 | 6,2322 | |
| 11 | $x_i$ | 1,51 | 1,55 | 1,57 | 1,59 | 1,64 | 1,72 | 1,9 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 12 | $x_i$ | 1,51 | 1,55 | 1,57 | 1,59 | 1,64 | 1,72 | 1,8 |
| | $y_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | |
| 13 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,338 |
| | $y_i$ | 3,281 | 3,521 | 3,634 | 3,782 | 3,801 | 3,921 | |
| 14 | $x_i$ | 5,4667 | 5,6802 | 5,5836 | 5,921 | 6,1308 | 6,2322 | 6,3 |
| | $y_i$ | 1,51 | 1,55 | 1,57 | 1,59 | 1,64 | 1,72 | |
| 15 | $x_i$ | 4,51 | 4,722 | 4,823 | 4,919 | 4,963 | 5,023 | 5,1 |
| | $y_i$ | 5,4667 | 5,6802 | 5,5836 | 5,921 | 6,1308 | 6,2322 | |
| 16 | $x_i$ | 4,51 | 4,722 | 4,823 | 4,919 | 4,963 | 5,023 | 5,05 |
| | $y_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | |
| 17 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,34 |
| | $y_i$ | 3,557 | 3,764 | 3,8 | 3,88 | 3,9875 | 4,005 | |
| 18 | $x_i$ | 4,51 | 4,722 | 4,823 | 4,919 | 4,963 | 5,023 | 5,2 |
| | $y_i$ | 1,51 | 1,55 | 1,57 | 1,59 | 1,64 | 1,72 | |

Окончание таблицы 14.1

| Номер варианта | Исходные данные | | | | | | | $x^*$ |
|---|---|---|---|---|---|---|---|---|
| 19 | $x_i$ | 4,51 | 4,722 | 4,823 | 4,919 | 4,963 | 5,023 | 5,1 |
| | $y_i$ | 3,281 | 3,521 | 3,634 | 3,782 | 3,801 | 3,921 | |
| 20 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,335 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 21 | $x_i$ | 1,51 | 1,55 | 1,57 | 1,59 | 1,64 | 1,72 | 1,79 |
| | $y_i$ | 3,281 | 3,521 | 3,634 | 3,782 | 3,801 | 3,921 | |
| 22 | $x_i$ | 4,51 | 4,722 | 4,823 | 4,919 | 4,963 | 5,023 | 5,05 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 23 | $x_i$ | 5,667 | 5,802 | 5,836 | 5,921 | 6,308 | 6,322 | 6,4 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 24 | $x_i$ | 0,2928 | 0,3103 | 0,3313 | 0,3476 | 0,3623 | 0,383 | 0,39 |
| | $y_i$ | 3,247 | 3,174 | 3,118 | 3,042 | 2,985 | 2,915 | |
| 25 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,35 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 26 | $x_i$ | 5,4667 | 5,6802 | 5,5836 | 5,921 | 6,1308 | 6,2322 | 6,4 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 27 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,335 |
| | $y_i$ | 3,281 | 3,521 | 3,634 | 3,782 | 3,801 | 3,921 | |
| 28 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,41 |
| | $y_i$ | 3,2557 | 3,1764 | 3,1218 | 3,0482 | 2,9875 | 2,9195 | |
| 29 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,38 |
| | $y_i$ | 3,247 | 3,174 | 3,118 | 3,042 | 2,985 | 2,915 | |
| 30 | $x_i$ | 0,298 | 0,303 | 0,31 | 0,317 | 0,323 | 0,33 | 0,335 |
| | $y_i$ | 5,4667 | 5,6802 | 5,5836 | 5,921 | 6,1308 | 6,2322 | |

# 15 Лабораторная работа № 15. Основы работы в среде R

## *15.1 Теоретические сведения*

Система статистического анализа и визуализации данных R состоит из следующих основных частей:

– языка программирования высокого уровня R, позволяющего одной строкой реализовать различные операции с объектами, векторами, матрицами, списками и т. д.;

– большого набора функций обработки данных, собранных в отдельные пакеты (package);

– развитой системы поддержки, включающей обновление компонентов среды, интерактивную помощь и различные образовательные ресурсы, предназначенные как для начального изучения R, так и последующих консультаций по возникающим затруднениям.

Статистическая среда R выполняет любой набор осмысленных инструкций языка R, содержащихся в файле скрипта или представленных последовательностью команд, задаваемых с консоли.

Консоль R представляет собой диалоговое окно, в котором пользователь вводит команды и где видит результаты их выполнения. Это окно возникает сразу при запуске среды. Кроме того, стандартный графический пользовательский интерфейс R (RGui) включает окно редактирования скриптов и всплывающие окна с графической информацией (рисунками, диаграммами и пр.).

При работе с использованием RGui рекомендуется создавать файл со скриптом (т. е. последовательностью команд языка R, выполняющей определенные действия). Как правило, это обычный текстовый файл с любым именем с расширением *.r, который можно создавать и редактировать обычным редактором типа «Блокнот». Если этот файл существует, его лучше всего поместить в рабочий каталог, и тогда после запуска R и выбора пункта меню «Файл > Открыть скрипт» содержимое этого файла появится в окне «Редактор R». Выполнить последовательность команд скрипта можно из пункта меню «Правка > Запустить все». Можно также выделить мышью осмысленный фрагмент из любого места подготовленного скрипта (от имени одной переменной до всего содержимого) и осуществить запуск этого блока на выполнение.

R обладает встроенными обширными справочными материалами, которые можно получить непосредственно в RGui. Если подать с консоли команду **help.start()**, то в интернет-браузере откроется страница, открывающая доступ ко всем справочным ресурсам: основным руководствам, авторским материалам, ответам на вероятные вопросы, спискам изменений, ссылкам на справки по другим объектам R и т. д.

Справку по отдельным функциям можно получить с использованием следующих команд:

– **help("foo")** или **? foo** – справка по функции **foo** (кавычки необязательны);

– **help.search("foo")** или **?? foo** – поиск всех справочных файлов, содержащих **foo**;

– **example("foo")** – примеры использования функции **foo**;

– **RSiteSearch("foo")** – поиск ссылок в онлайн-руководствах и архивах рассылок;

– **apropos("foo", mode="function")** – список всех функций с комбинацией **foo**;

– **vignette("foo")** – список руководств по теме **foo**.

Удобным средством освоения вычислений в R является R Commander – платформо-независимый графический интерфейс в стиле кнопочного меню, реализованный в пакете Rcmdr. Он позволяет осуществить большой комплект процедур статистического анализа, не прибегая к предварительному заучиванию

функций на командном языке. Установить Rcmdr, как и любые другие расширения, можно из меню консоли R «Пакеты > Установить пакет», но лучше, выполнив команду **install.packages("Rcmdr", dependencies=TRUE)**, где включение опции dependencies вызовет гарантированную установку полного комплекта остальных пакетов, которые могут потребоваться при обработке данных через меню Rcmdr. Запуск R Commander происходит при загрузке пакета Rcmdr через меню «Пакеты > Включить пакет» или командой **library(Rcmdr).** Если по какой-то причине было принято решение анализировать данные исключительно с помощью R Commander, то для автоматической загрузки этой графической оболочки при запуске R необходимо отредактировать файл Rprofile.site.

R Commander позволяет выполнить первичный анализ данных, достаточно большое количество статистических тестов для имеющихся данных, а также широкие графические возможности. Загрузить или импортировать данные можно с помощью меню «Данные», выполнить статистические тесты или получить статистические оценки набора данных с помощью меню «Статистика», получить графические представления с помощью меню «Графики».

### *Контрольные вопросы*

1 Каково основное назначение пакета R?
2 Перечислите этапы обработки данных.
3 Опишите назначение пакета Rcmdr.

### *15.2 Задания для самостоятельного выполнения*

1 Запустите R Commander.
2 Загрузите набор данных из пакета datasets HairEyeColor.
3 С помощью агрегации переменных активного датасета сформируйте наборы данных, содержащие информацию о распределении наблюдаемых данных по половому признаку, по цвету волос и по цвету глаз.
4 С помощью вкладки «Графики» постройте точечные графики для каждого агрегированного набора данных.
5 С помощью вкладки «Графики» постройте графики средних частот для сочетаний признаков «Пол» – «Цвет волос», «Пол» – «Цвет глаз», «Цвет волос» – «Цвет глаз». Сделайте выводы о наиболее часто встречающихся сочетаниях.

# 16 Лабораторная работа № 16. Основы программирования на языке R

## *16.1 Теоретические сведения*

R представляет собой язык программирования высокого уровня.

Все объекты данных (в том числе и переменные) в R можно разделить на следующие классы (типы объектов):

– **numeric** – объекты, к которым относятся целочисленные (**integer**) и действительные (**double**) числа;

– **logical** – объекты, которые принимают только два значения: **False** (**F**) и **True** (**T**);

– **character** – объекты, символьные значения, которые обычно задаются в двойных либо одинарных кавычках.

В R допускается давать имена переменным как на латинице, так и на кириллице с учетом чувствительности к регистру. Идентификаторы должны начинаться с буквы или точки и состоять из букв, цифр, знаков препинания и подчеркивания. При помощи команды **? <имя>** можно проверить, существует ли переменная или функция с указанным именем.

Проверка на принадлежность переменой определенному классу проверяется функциями **is.numeric()**, **is.integer()**, **is.logical()**, **is.character()**, для преобразования объекта в другой тип можно использовать функции **as.numeric()**, **as.integer()**, **as.logical()**, **as.character()**.

В R существует ряд специальных объектов:

– **Inf** – положительная или отрицательная бесконечность;

– **NA** – «отсутствующее значение»;

– **NaN** – «не число».

Проверить, является ли объект принадлежащим к одному из специальных типов можно с помощью функций **is.finite()**, **is.na()**, **is.nan()**.

*Выражение* (*expression*) языка R представляет собой сочетание таких элементов, как оператор присваивания, арифметические или логические операторы, имена объектов и имена функций. Результат выполнения выражения, как правило, сразу отображается в командном или графическом окне. Однако при выполнении операции присваивания результат сохраняется в соответствующем объекте и на экран не выводится.

В качестве оператора присваивания в R можно использовать либо символ “=“, либо пару символов “<–“ (присваивание определенного значения объекту слева) или “–>“ (присваивание значения объекту справа). Хорошим стилем программирования считается использование “<–“.

Выражения языка R организуются в скрипте по строкам. В одной строке можно ввести несколько команд, разделяя их символом “;“. Одну команду можно также расположить на двух (и более) строках.

Объекты типа **numeric** могут составлять выражения с использованием традиционных арифметических операций + (сложение), – (вычитание),

* (умножение), / (деление), ^ (возведение в степень), %/% (целочисленное деление), %% (остаток от деления). Операции имеют обычный приоритет, т. е. сначала выполняется возведение в степень, затем умножение или деление, потом уже сложение или вычитание. В выражениях могут использоваться круглые скобки и операции в них имеют наибольший приоритет.

Логические выражения могут составляться с использованием следующих логических операторов:

– «Равно» ==;
– «Не равно» !=;
– «Меньше» <;
– «Больше» >;
– «Меньше либо равно» <=;
– «Больше либо равно» >=;
– «Логическое И; &;
– «Логическое ИЛИ» |;
– «Логическое НЕ» !.

**Вектор** представляет собой поименованный одномерный объект, содержащий набор однотипных элементов (числовые, логические либо текстовые значения – никакие их сочетания не допускаются). Для создания векторов небольшой длины в R используется функция конкатенации **c()** (от «*concatenate*» – объединять, связывать). В качестве аргументов этой функции через запятую перечисляют объединяемые в вектор значения. Вектор можно создать также при помощи функции **scan()**, которая «считывает» последовательно вводимые с клавиатуры значения. В случае ошибочного ввода необходимо корректировать данные в режиме редактирования.

Для создания векторов, содержащих последовательную совокупность чисел, удобна функция **seq()** (от «*sequence*» – последовательность) с указанием начала и конца диапазона. Аналогичный результат можно получить, указав в операторе присваивания начало и конец диапазона через символ двоеточие «:». В качестве дополнительного параметра функция **seq()** принимает шаг приращения. Векторы, содержащие повторяющиеся значения, создают с помощью функции **rep**() с указанием необходимого количества повторений.

Векторы можно конкатенировать независимо от типов хранящихся в них данных с помощью функции **c()**.

Каждому элементу вектора при создании присваивается номер (индекс), начиная с 1. Для обращения к нужному элементу вектора необходимо вызвать его указанием имени вектора и номера элемента в квадратных скобках. Также в квадратных скобках через двоеточие можно указывать индексы срезов элементов, обращаясь сразу к нескольким элементам вектора. Допускается обращение к нескольким не идущим подряд элементам с указанием их индексов в функции конкатенации через запятую. Критерием выбора элементов вектора может являться логическое выражение.

Для сортировки элементов векторов применяют функцию **sort**() с указанием аргумента **decreasing = False** или **True** для сортировки по возрастанию или убыванию соответственно.

Матрица представляет собой двумерный вектор. Для создания матрицы используется функция **matrix**(), принимающая в качестве аргументов набор элементов (вектор) и параметры, определяющие количество срок и столбцов, при этом заполнение матрицы происходит по столбцам. Порядок заполнения можно изменить, присвоив параметру **byrow** значение **True**.

В качестве заголовков строк и столбцов создаваемой матрицы автоматически выводятся индексные номера, но можно присвоить им пользовательские значения с помощью функций **rownames()** и **colnames()**.

Сформировать матрицу на основе имеющегося вектора возможно с помощью функции **dim()**, присвоив вектор значений размерностей.

Также можно сформировать матрицу из нескольких векторов, используя функции **cbind()** или **rbind()** (сцепкой столбцов или строк).

В статистике данные очень часто группируют в соответствии с тем или иным признаком, например, полом, социальным положением, стадией болезни, местом отбора проб и т. п. В R существует специальный класс векторов – **факторы** (*factors*), которые предназначены для хранения кодов соответствующих *уровней* номинальных признаков. Часто уровни факторов кодируют в виде чисел. В таких случаях очень важно «проинструктировать» программу так, чтобы она «распознавала» уровни номинальной переменной от чисел как таковых. Для этого используется функция **factor()**, которая на основе набора данных (вектора) строит фактор, группирующий полученные данные по уровням фактора. Уровни фактора можно указать в параметре levels перечислением их в векторе.

Также полезным свойством R является то, что уровни фактора можно кодировать не только числовыми данными, но и текстовыми.

Существует специальная команда для создания факторов

**gl(n, k, length, labels)**,

принимающая в качестве параметров n – количество факторов; k – количество значений каждого фактора; length – размер итогового объекта; labels – имена для каждого уровня фактора.

Команда **cut(x, breaks, labels)** создает факторы, разделив область вариации числового вектора x на интервалы.

Для сохранения разнородной информации в R реализованы списки (**list**) и таблицы данных (**data frame**). Каждый компонент этих объектов может быть списком, матрицей, вектором, фактором.

Для создания списков служит команда **list()**, компонентам которого присваиваются имена. Доступ к элементам списков может быть по их именам с указанием их через символ «$» с помощью индексации, причем первый индекс (для доступа к компоненту списка) необходимо указывать в двойных квадратных скобках.

Если структура списка неизвестна, получить информацию о ней можно с помощью функции **str()**, которая отображает подробную структуру списка и его элементов.

**Таблица** данных (*data frame*) представляет собой объект R, по структуре напоминающий лист электронной таблицы Microsoft Excel. Каждый столбец

таблицы является вектором, содержащим данные определенного типа. При этом действует правило, согласно которому все столбцы должны иметь одинаковую длину (собственно, с «точки зрения» R таблица данных является частным случаем списка, в котором все компоненты – векторы имеют одинаковый размер).

Таблицы данных – это основной класс объектов R, используемых для хранения данных. Обычно такие таблицы подготавливаются при помощи внешних приложений (особенно популярна и удобна программа Microsoft Excel) и затем загружаются в среду R. Небольшую таблицу можно собрать из нескольких векторов средствами самой системы R. Для этого используют функцию **data.frame()**. При этом синтаксис добавляемых структур данных описывается в формате «заголовок столбца = добавляемый вектор». В качестве заголовков столбцов могут выступать любые пользовательские имена, удовлетворяющие требованиям R.

Извлечь отдельные компоненты таблиц для выполнения необходимых вычислений, как и в примерах со списками, можно с использованием знака $, квадратных скобок с указанием двух индексов [<номер_строки>, номер_столбца>], двойных квадратных скобок [[]] либо непосредственно по имени столбца. После имени или индексного номера столбца можно указывать индексные номера отдельных ячеек таблицы, что позволяет извлекать содержимое этих ячеек.

Отобразить информацию о структуре таблицы данных можно с помощью функции **str()**. Вывести имена переменных можно при помощи команды **names()**.

Имеется также возможность быстро просмотреть несколько первых или несколько последних значений каждой переменной, входящей в состав таблицы данных. Для этого используются функции **head()** и **tail()** соответственно.

Часто на практике некоторые значения в таблице отсутствуют. Ячейки с такими отсутствующими значениями (*missing values*) в таблицах данных R не могут быть просто пустыми – иначе столбцы таблицы окажутся разной длины. Для обозначения отсутствующих наблюдений в языке R, как указывалось ранее, имеется специальное значение – **NA** (*not available* – не доступно). Если значение **NA** имеет смысл нуля (например, экземпляров некого вида обнаружено не было), то легко произвести эту замену в таблице DF командой **DF[is.na(DF)] <– 0**.

Для сортировки строк таблицы по различным ключам применяется функция **order()**.

Функции в R представляют собой поименованный программный код, состоящий из некоторого набора переменных, констант, операторов и других функций и предназначенный для выполнения конкретных операций и задач. Как правило (но не всегда), функции возвращают результат своего выполнения в виде объекта языка R – переменной определенного класса: вектора, списка, таблицы и т. д.

По своему назначению функции можно разделить на характерные группы: арифметические, символьные, статистические и пр. Функции могут быть встроенными (т. е. представленными в базовых или подгружаемых пакетах) и собственными (т. е. написанными непосредственно самими пользователями). Некоторые наиболее употребительные встроенные функции представлены далее:

– **abs(x)** – модуль величины x;

– **ceiling(x)** – округление до целого в большую сторону;

– **floor(x)** – округление до целого в меньшую сторону;

– **round(x, digits = n)** – округление числа x до n знаков после десятичной точки;

– **signif(x, digits = n)** – округление до указанного числа значащих цифр;

– **trunc(x)** – округление до целого числа;

– **exp(x)** – вычисление экспоненты;

– **log(x)** – логарифм натуральный;

– **log10(x)** – логарифм десятичный;

– **sqrt(x)** – корень квадратный;

– **cos(x)**, **sin(x)**, **tan(x)**, **acos(x)**, **cosh(x)**, **acosh(x)** – тригонометрические функции;

– **grep(pattern, x)** – возвращает индекс первого найденного элемента pattern в x;

– **substr(x, start, stop)** – выбор или замена символов в строках символьного вектора x;

– **paste(…, sep= “”)** – объединение символов или строк через значение разделителя;

– **toupper(x)**, **tolower(x) –** преобразование текстового вектора x в прописные и обратно.

Одной из особенностей языка программирования R является модульность. Под модульностью понимается использование групп выражений и функций, которые группируются внутри фигурных скобок. Результатом выполнения группы является последнее записанное в фигурных скобках выражение. Для удобства такие группы выражений оформляются в виде функций, синтаксис объявления которых таков:

**имя_функции <– function(arg1, arg2, …) {**
**группа выражений**
**return(object) }**

Здесь имя_функции – имя создаваемой функции; arg1, arg2, … – формальные аргументы функции. Оператор **return()** нужен в случаях, когда группа выражений не возвращает целевого результата.

Перед первым выполнением функция должна быть определена в текущем скрипте или загружена из внешнего файла, где она была предварительно подготовлена. Тогда вызов функции может быть осуществлен как

**имя_функции (arg1, arg2, …)**

Как и в любом алгоритмическом языке, в R широко используются ветвления и циклы. Условный оператор имеет следующую структуру:

**if { логическое_выражение }**
**{ группа_выражений если True }**
**else {группа выражений если False }**

Имеется также сокращенная форма реализации ветвлений:

**ifelse (логическое_выражение, группа_выражений_1, группа_выражений_2)**

Повторение в цикле одних и тех же выражений или вычислений осуществляется с помощью циклических конструкций **for**, **while** или **repeat**:

**for (index in for_object) {группа_выражений}**

**while (логическое_выражение) {группа_выражений}**

**repeat {группы_выражений; break}**

Объект for_object может быть вектором, массивом, таблицей или списком.

Язык программирования R поддерживает векторизацию вычислений, которая заключается в том, что вместо выполнения скалярных операций с каждым элементом вектора или массива эти операции выполняются параллельными вычислениями, причем обрабатываются все элементы массива одновременно. Такой подход позволяет существенно снизить время вычислений. В R есть целое семейство функций, предназначенных для векторизованных вычислений. В названии таких функций имеется свойство **apply**, с принципом работы которого можно ознакомиться в справочном руководстве. Аналогичная функция **lapply()** позволяется применить какую-либо функцию к каждому компоненту списка и получить результат также в виде списка.

### *Контрольные вопросы*

1 В чем состоит концепция модульности в языке программирования R?

2 В чем состоит концепция векторизации вычислений в языке программирования R?

3 Опишите синтаксис условного оператора в языке программирования R.

4 Опишите синтаксис циклического оператора for в языке программирования R.

5 Опишите синтаксис циклического оператора while в языке программирования R.

6 Опишите синтаксис циклического оператора repeat в языке программирования R.

### *16.2 Задания для самостоятельного выполнения*

1 Напишите программу на языке программирования R, которая в заданном числовом векторе все значения, большие 10, заменяет их квадратами.

2 Напишите программу на языке программирования R, которая переменной my_age присваивает числовое значение, соответствующее Вашему возрасту, и выводит сообщение о том, являетесь ли Вы совершеннолетним.

3 Напишите программу на языке программирования R, которая с помощью команды scan с клавиатуры получает вектор group_height значений роста студентов Вашей группы, выводит вектор значений group_height на экран, вычисляет среднее значение роста по группе и сохраняет его в переменной ave_height, выводит это значение, вычисляет отклонение значения роста студентов группы от среднего значения, сохраняет полученные значения в вектор delta, выводит его на экран.

4 Напишите программу на языке программирования R, которая создает две матричных переменных – *a* и *b*, заполняет каждую значениями из последовательных чисел от 1 до 9, при этом матрица *a* – это матрица-столбец, а матрица *b* – матрица-строка; создает матрицу *c*, полученную произведением *a* и *b*; задает имена срокам и столбцам матрицы *c*, установив их соответственно последовательными числами от 1 до 9; выводит результаты счета на экран.

5 Напишите программу на языке программирования R, которая получает с клавиатуры вектор vote с результатами голосования Вашей учебной группы по поводу любимого цвета из имеющихся двух цветов – «красный» и «черный»; создает фактор colors_factor; формирует таблицу colors_stat, отражающую количество вхождений каждого фактора; выводит таблицу colors_stat на экран.

6 Напишите программу на языке программирования R, которая с клавиатуры получает фамилии студентов Вашей группы, результаты голосования о выборе цвета по схеме «красное», «черное», рост студентов группы; объединяет векторы names, vote, height в таблицу данных my_stats с помощью команды data.frame(); устанавливает названия компонентов соответственно Names, Color, Height; с помощью команды fix() и встроенного редактора позволяет добавить к набору данных фактор, содержащий значения «Сидорова», «черный», 166; выводит на экран набор данных my_stats; выводит на экран фамилии людей, рост которых превышает 170 см.

7 Ознакомьтесь с разделом справки, связанным с функцией as.POSIXlt(). Напишите программу на языке программирования R, которая создает переменную *date*1, содержащую информацию о текущей дате, переменную *date*2, содержащую информацию о Вашей дате рождения; выводит на экран переменные *date*1, *date*2, количество прожитых Вами дней, лет.

# 17 Лабораторная работа № 17. Базовые графические возможности R

## *17.1 Теоретические сведения*

Для работы с графикой в R используются функции высокого уровня, в которых определяется размерность графика, масштабы осей, названия и др. Наиболее часто употребляемая для построения графиков функция **plot()**. Используется функция в следующем формате:

**plot(x, y, …)**.

Для ее вызова достаточно передать в функцию два вектора, определяющие независимую переменную и значения функции. Для более тонкой настройки следуем использовать параметры:

– **type** – тип графика, возможные значения «p» для точечного графика, «l» для линейного графика, «b» включает оба предыдущих типа одновременно и т. д.;

– **main** – заголовок графика;

– **sub** – подназвание для графика;

– **xlab**, **ylab** – названия осей;

– **asp** – соотношение осей;

– **xlim**, **ylim** – параметры, управляющие ограничениями значений по осям, принимают значения в виде числового вектора;

– **axes**, **ann** – контролируют отображение осей и их названий, принимают логические значения;

– **log** – переводит одну из осей или обе одновременно в формат логарифмической;

– **pch** – числовой параметр, тип значка на графике, принимает значения от 1 до 25.

Гистограмма по имеющимся данным строится функцией **hist()**.

Диаграмму размахов можно отобразить, используя функцию **boxplot()**.

Вывод круговой диаграммы организован с помощью функции **pie()**.

Вместить в одно изображение позволяет функция **par()**.

Более подробно с параметрами функций можно ознакомиться в справочной системе R.

### *Контрольные вопросы*

1 Перечислите основные функции для вывода графической информации в R.

## *17.2 Задания для самостоятельного выполнения*

1 Напишите программу на языке программирования R, которая генерирует случайный набор данных (функция rnorm()), моделирующих рост 100 человек, с параметрами mean = 170, sd = 30, строит на одном графике гистограмму плотности вероятности распределения случайного вектора $X$, сглаженный график его плотности вероятности, график его плотности вероятности без сглаживания. Цвет гистограммы – светло-голубой, цвет графика плотности вероятности – красный, цвет несглаженного графика плотности вероятности – синий, толщина графиков – 2.

2 Напишите программу на языке программирования R, которая с клавиатуры получает вектор vote, состоящий из результатов голосования в текстовой форме (да, нет), размерность которого не менее 20 измерений; строит круговую диаграмму, отражающую результаты голосования.

# Список литературы

1 **Карманов, Ф. И.** Статистические методы обработки экспериментальных данных с использованием пакета MathCad: учебное пособие / Ф. И. Карманов, В. А. Острейковский. – Москва : КУРС ; ИНФРА-М, 2019. – 208 с.

2 **Плохотников, К. Э.** Базовые разделы математики для бакалавров в среде MATLAB: учебное пособие / К. Э. Плохотников. – 2-е изд. – Москва: ИНФРА-М, 2018. – 1114 с.

3 **Волкова, П. А.** Статистическая обработка данных в учебно-исследовательских работах : учебное пособие / П. А. Волкова, А. Б. Шипунов. – Москва : ФОРУМ ; ИНФРА-М, 2020. – 96 с.