ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования
Московский технический университет связи и информатики

В.Н.Шакин, Т.И.Семенова

Основы работы с математическим пакетом Matlab

Учебное пособие

Москва 2015

ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования
Московский технический университет связи и информатики

В.Н.Шакин, Т.И.Семенова

Основы работы с математическим пакетом Matlab

Учебное пособие

для направления

11.03.02 – Инфокоммуникационные технологии и системы связи

Москва 2015

УДК 32.973.26018.2

Шакин В.Н., Семенова Т.И. Основы работы с математическим пакетом Matlab. Учебное пособие / МТУСИ. – М., 2015. – 133 с.

Учебное пособие посвящено для работы с пакетом Matlab, являющимся на настоящий момент одной из самых современных математических пакетов.

В первом разделе пособия, достаточно полно описываются элементы рабочей среды последних версий Matlab.

Второй раздел пособия составляет материал, посвященный технологиям решения средствами Matlab задач вычислительной математики: аппроксимация и интерполяция функций, решение нелинейных и дифференциальных уравнений, оптимизация функций.

Ил. 119, табл. 22, список лит 4 назв.

Издание утверждено советом факультета .
Протокол № от

Рецензенты: М.Я.Клепцов, д.т.н., профессор

Содержание

Введение…… 5
Раздел 1. Основы работы с математическим пакетом Matlab…… 11
Тема 1.1. Элементы рабочей среды Matlab…… 11
1.1.1. Элементы рабочей среды Matlab и простейшие вычисления……... 11
1.1.2. Основные объекты системы Matlab……... 26
1.1.3. Лабораторная работа по теме……. 36
1.1.4. Контрольные вопросы по теме …….. 38

Тема 1.2. Векторы, матрицы и построение графиков в системе Matlab…….. 39
1.2.1. Векторы и матрицы…… 39
1.2.2. Построение графиков и визуализация вычислений в системе Matlab……. 47
1.2.3. Лабораторная работа по теме……..……. 57
1.2.4. Контрольные вопросы по теме……. 61

Тема 1.3. Средства Matlab для создания и описания m-файлов…….…….... 62
1.3.1. Основные понятия и средства программирования в Matlab…… 62
1.3.2. Описание и работа со script-файлами……. 64
1.3.3. Описание и работа с m-функциями…… 67
1.3.4. Алгоритмические операторы Matlab ……. …….. 70
1.3.5. Примеры решения задач с использованием m-файлов.. 81
1.3.6. Лабораторная работа по теме ……. 84
1.3.7. Контрольные вопросы по теме……. 89

Раздел 2. Технология решения вычислительных задач средствами Matlab…… 90
Тема 2.1. Решение нелинейных уравнений…… 90
2.1.1. Численное решение нелинейных уравнений…….. 90
2.1.2. Лабораторная работа по теме ……. 94
2.1.3. Контрольные вопросы по теме…… 96

Тема 2.2. Технология аппроксимации и интерполяции функций в среде пакета Matlab…… 97
2.2.1. Аппроксимация и интерполяция функций…….. 97
2.2.2. Лабораторная работа по теме ……. 102
2.2.3. Контрольные вопросы по теме…… 104

Тема 2.3. Технология интегрирования в среде Matlab ……..... 105
2.3.1. Вычисление неопределенных и определенных интегралов……. 105
2.3.2. Лабораторная работа по теме ……. 109
2.3.3. Контрольные вопросы по теме…… 112

Тема 2.4. Технология решения обыкновенных дифференциальных уравнений……..……. 113
2.4.1. Численное решение обыкновенных дифференциальных уравнений…… 113
2.4.2. Лабораторная работа по теме……..... 118
2.4.3. Контрольные вопросы по теме…… 120

Тема 2.5. Технология решения задач одномерной оптимизации…….. 121
2.5.1. Решение задач одномерной оптимизации ……. 121
2.5.2. Лабораторная работа по теме……. 124
2.5.3. Контрольные вопросы по теме …… 125

Тема 2.6. Технология решения задач многомерной оптимизации…… 126
2.6.1. Решение задач многомерной оптимизации…… 126
2.6.2. Лабораторная работа по теме……..……... 129
2.6.3. Контрольные вопросы по теме……... 131

Список литературы…….. 131

Введение

Компьютер или персональный компьютер (ПК) является универсальным устройством для обработки информации. Однако сам по себе ПК не обладает знаниями ни в одной области своего применения, все эти знания сосредоточены в программном обеспечении (ПО) – программах, выполняемых на нем.

Все ПО принято делить на ***системное*** и ***пользовательское (прикладное)***. Системное программное обеспечение выполняет функции «управления» всех элементов ПК, а пользовательские программы служат для выполнения конкретных задач, возникающих во всех сферах человеческой деятельности для конкретных пользователей.

Многоуровневое представление ПК – модель представления ПК в виде совокупности взаимосвязанных уровней, разделенных по функциональному назначению (рис. В-1).

Рис. В-1. Взаимодействие между пользователем, прикладным программным обеспечением, операционной системой и аппаратным обеспечением

Пользователь – это человек, принимающий участие в управлении объектами и системами некоторой предметной области и являющийся составным элементом автоматизированной системы.

Программа – последовательность формализованных инструкций, представляющих алгоритм решения некоторой задачи, предназначенная для исполнения вычислительной машины.

Операционная система (ОС) – это комплекс взаимосвязанных системных программ, выполняющий роль связующего звена между аппаратурой компьютера, выполняемыми программами и пользователем. ОС обычно хранится во внешней памяти компьютера – на диске, а при включении компьютера считывается с дисковой памяти и

размещается в ОЗУ. Этот процесс называется загрузкой операционной системы.

Программное обеспечение (ПО) – совокупность программ и данных, предназначенных для решения определенного круга задач, хранящихся на машинных носителях. ПО, в свою очередь, подразделяется на ***системное*** (совокупность программ и программных комплексов для обеспечения работы компьютера) и ***прикладное*** (пакеты прикладных программ и интегрированные программные системы, предназначенные для решения различных задач пользователей: ***математическое, офисное, графическое*** *и т.п.*).

Системное ПО подразделяется на **базовое** и **сервисное**. **Базовое ПО** включает в себя операционные системы, оболочки, сетевые операционные системы, а **сервисное ПО** – программы (утилиты): диагностики, антивирусные, обслуживания носителей, архивирования и обслуживания сети.

Особую группу составляют ***системы программирования***, которые являются частью системного ПО, но носят прикладной характер. Это совокупность программ для разработки, отладки и внедрения новых программных продуктов. Системы программирования обычно содержат: трансляторы; среду разработки программ; библиотеки программ; отладчики; редакторы связей и др.

Соотношение между требующимися программными продуктами и имеющимися на рынке меняется очень быстро. На сегодняшний день сложились следующие группы программного обеспечения: ***операционные системы и оболочки; системы программирования (трансляторы, библиотеки подпрограмм, отладчики и т.д.); инструментальные системы; интегрированные пакеты программ; системы машинной графики; системы управления базами данных (СУБД); прикладное программное обеспечение.***

Из-за огромного разнообразия ***прикладного программного обеспечения*** (**ППО**) существует множество вариантов его классификации. Наиболее общая классификация предполагает разделение **ППО** на два основных класса:

- прикладные программы ***общего назначения***, к которым относятся программы, обеспечивающие выполнение наиболее часто используемых, универсальных задач (текстовые редакторы, табличные процессоры, графические редакторы и т.п.);
- прикладные программы ***специального (профессионального) назначения***, к которым относятся программы, ориентированные на достаточно узкую предметную область (издательские системы, САПР - системы автоматизированного проектирования, банковские, бухгалтерские программы, программы 3D-графики, программы видеомонтажа, нотные редакторы и т.д.).

Библиотека стандартных подпрограмм (процедур, методов) – это совокупность подпрограмм, составленных на одном из языков программирования и удовлетворяющих определенным единым требованиям к структуре, организации их входов и выходов, описаниям подпрограмм и т.п. **Стандартные подпрограммы** имеют единую форму обращения, что обеспечивает простоту и удобство настройки параметров подпрограммы на решение конкретной задачи.

Пакеты прикладных программ (ППП) – это специальным образом организованные программные комплексы, рассчитанные на общее применение в определенной проблемной области и дополненные соответствующей технической документацией. В зависимости от характера решаемых задач различают: пакеты для решения типовых инженерных, планово-экономических, общенаучных задач; пакеты системных программ; пакеты для обеспечения систем автоматизированного проектирования и систем автоматизации научных исследований; пакеты обучающих программных средств и другие.

Интегрированные пакеты представляют собой набор нескольких программных продуктов, объединенных в единый удобный инструмент. Наиболее развитые из них включают в себя текстовый редактор, органайзер, электронную таблицу, **СУБД**, средства поддержки электронной почты, программу создания презентационной графики. Интегрированные пакеты, как правило, содержат некоторое **ядро**, обеспечивающее возможность тесного взаимодействия между составляющими.

Наиболее известным интегрированным пакетом является **MS Office.** В этот мощный профессиональный пакет вошли такие необходимые программы, как текстовый редактор **MS Word**, табличный процессор **MS Excel**, программа создания презентаций **Power Point, СУБД Access** и средство поддержки электронной почты **E-mail.** Все части этого пакета составляют единое целое, и даже внешне все программы выглядят единообразно, что облегчает как их освоение, так и ежедневное использование.

Для инженерных и научных расчетов используются ***универсальные математические пакеты символьной и численной математики***, такие, например, как **Mathematic, MathCAD, Maple и MatLab**.

Итак, ***пакет прикладных программ (ППП)*** – это комплекс взаимосвязанных программ для решения определенного класса задач из конкретной предметной области. На текущем этапе развития информационных технологий именно ППП являются наиболее востребованным видом прикладного обеспечения. Это связано с особенностями ППП. Рассмотрим их подробней.

- ***Ориентация на решение класса задач,*** заключающаяся в ориентации ППП не на отдельную задачу, а на некоторый класс задач, в том числе и специальных, из определенной предметной

области. Так, например, офисные пакеты ориентированы на офисную деятельность, следовательно, он должен реализовывать функции обработки текста, иметь средства обработки табличной информации, средства построения диаграмм разного вида и первичные средства редактирования растровой и векторной графики.

- ***Наличие языковых средств***, позволяющее расширить число задач, решаемых пакетом или адаптировать пакет под конкретные нужды. Поддерживаемые языки могут быть использованы для формализации исходной задачи, описания алгоритма решения и начальных данных, организации доступа к внешним источникам данных, разработки программных модулей, описания модели предметной области, управления процессом решения в диалоговом режиме и других целей.
- ***Единообразие работы с компонентами пакета,*** состоящее в наличии специальных системных средств, обеспечивавших унифицированную работу с компонентами. К их числу относятся специализированные банки данных, средства информационного обеспечения, средства взаимодействия пакета с операционной системой, типовой пользовательский интерфейс и т.п.

Несмотря на разнообразие конкретных пакетных разработок, их обобщенную внутреннюю структуру можно представить в виде трех взаимосвязанных элементов (рис. В-2):

1) ***входной язык*** (макроязык, язык управления) – средство общения пользователя с пакетом (это как универсальные (Pascal, Basic и т.п.), так и специализированные, проблемно-ориентированные языки программирования, развитый пакет может обладать несколькими входными языками);
2) ***предметное обеспечение*** (функциональное наполнение) – реализует особенности конкретной предметной области и программные модули, реализующие алгоритмы (или их отдельные фрагменты) прикладных задач;
3) ***системное обеспечение*** (системное наполнение) – низкоуровневые средства, например, доступ к функциям операционной системы, например, программа, управляющая взаимодействием всех компонентов ППП; транслятор(ы) с входных языков; средства доступа к данным; информационно-справочный модуль; различные служебные программы и т.д.

Рис. В-2. Структура ППП

Приведенная логическая структура ППП достаточно условна, поскольку в конкретном ППП может отсутствовать четкое разделение программ на предметное и системное обеспечение.

Современные ППП, такие как **Maple, Mathematica, Matlab** и некоторые другие, являются сложными программными системами, включающими специализированные системные и языковые средства. Хотя синтаксис языка пользователя у них различный и библиотеки доступных функций могут меняться от нескольких сотен до тысяч, да и внутренние структуры и даже используемые алгоритмы значительно отличаются друг от друга, все они обладают общими свойствами. Таких принципиальных общих свойств значительно больше, чем различий и таким образом после освоения одной из систем компьютерной алгебры переход к другой системе не является сложной проблемой.

Каждый математический пакет имеет особенности в своей архитектуре или структуре. Тем не менее, для современных универсальных математических пакетов можно представить общую структуру, которая состоит из следующих элементов.

- ***Ядро системы*** – программные коды множества заранее откомпилированных функций и процедур, обеспечивающих достаточно представительный набор встроенных функций и операторов системы.
- ***Интерфейс*** дает пользователю возможность обращаться к ядру со своими запросами и получать результат решения на экране дисплея;
- ***Библиотеки*** функций и процедур, включенные в ядро, выполняются предельно быстро.
- ***Пакеты расширения*** позволяют осуществить расширение возможностей систем и их адаптацию к решаемым конкретными пользователями задачам.
- ***Справочная система*** аккумулирует знания в области математики.

Математический пакет Matlab предназначен как для численных вычислений, так и для применения в сфере символьной

математики и моделирования. ***Matlab*** – это одна из старейших, тщательно проработанных и проверенных временем систем автоматизации математических расчетов, построенная на расширенном представлении и применении матричных операций. Это нашло отражение и в самом названии системы – ***MATrix LABoratory***, то есть матричная лаборатория. Однако синтаксис языка программирования системы продуман настолько тщательно, что данная ориентация почти не ощущается теми пользователями, которых не интересуют непосредственно матричные вычисления.

Несмотря на то, что изначально Matlab предназначался исключительно для вычислений, в процессе эволюции (а сейчас выпущена уже версия 2015a), в дополнение к численным средствам, в Matlab был разработан пакет символьных преобразований, а также появились библиотеки, которые обеспечивают в Matlab уникальные для математических пакетов функции.

В системе Matlab также существуют широкие возможности для программирования. Ее библиотека является **объектной** (объектно-ориентированной) и содержит большое количество процедур обработки данных на языке С. Внутри пакета можно использовать как процедуры самой системы Matlab, так и стандартные процедуры языка С, что делает этот инструмент мощнейшим подспорьем при разработке приложений (используя компилятор С, можно встраивать любые процедуры Matlab в готовые приложения).

Все библиотеки Matlab отличаются высокой скоростью численных вычислений. Здесь матрицы широко применяются не только в таких математических расчетах, как решение задач линейной алгебры и математического моделирования, обсчета статических и динамических систем и объектов. Они являются основой автоматического составления и решения уравнений состояния динамических объектов и систем. Именно универсальность аппарата матричного исчисления значительно повышает интерес к системе Matlab, вобравшей в себя лучшие достижения в области быстрого решения матричных задач. Поэтому Matlab давно уже вышла за рамки специализированной матричной системы, превратившись в одну из наиболее мощных универсальных интегрированных систем компьютерной математики.

Для визуализации моделирования система Matlab имеет библиотеку, которая обеспечивает широкий спектр функций, поддерживающих визуализацию проводимых вычислений непосредственно из среды Matlab, увеличение и анализ, а также возможность построения алгоритмов обработки изображений. Усовершенствованные методы графической библиотеки в соединении с языком программирования Matlab обеспечивают открытую расширяемую систему, которая может быть использована для создания специальных приложений, пригодных для обработки графики.

Таким образом, система Matlab вобрала опыт, и правила, и методы математических вычислений, накопленные за тысячи лет развития математики. Одну только прилагаемую к системе обширную документацию

вполне можно рассматривать как фундаментальный многотомный электронный справочник по математическому обеспечению.

Раздел 1. Основы работы с математическим пакетом Matlab

Тема 1.1. Элементы рабочей среды Matlab

1.1.1. Элементы рабочей среды Matlab и простейшие вычисления
1.1.2. Основные объекты системы Matlab
1.1.3. Лабораторная работа по теме
1.1.4. Контрольные вопросы по теме

1.1.1.Элементы рабочей среды Matlab и простейшие вычисления

Графический интерфейс пользователя Matlab версий R2014 – R2015 аналогичен интерфейсу других Windows-приложений, имеющих ленточную структуру. После запуска Matlab на экране дисплея появляется **Рабочая среда** (интерфейс пользователя) системы Matlab в стандартной конфигурации (рис. 1.1.1-1). При этом система Matlab готова к проведению вычислений в командном режиме, т.е. в окне **Command Window**. Если окно **Рабочая среда** системы Matlab находится не в стандартной конфигурации, то необходимо щелкнуть мышкой по элементу **LAYOUT** инструментальной панели (рис. 1.1.1-1 выноска 15), в результате чего откроется окно **SELECT LAYOUT** (рис. 1.1.1-2), в котором необходимо выполнить команду **Default**. Команда **Default** возвращает внешний вид **Рабочей среды**, который был принят по умолчанию (рис. 1.1.1-1).

Рабочая среда системы Matlab – это обычное окно приложений MS Windows, поэтому его можно перемещать, изменять в размерах, открывать на весь экран. В этом окне стандартной конфигурации могут быть размещены следующие компоненты (на рис. 1.1.1-1 отображены соответствующими выносками, пронумерованными от 1 до 10, компоненты, пронумерованные 11-14 показаны на соответствующих рисунках):

1) **Стандартное меню** позволяет выполнить команды **Сохранить, Вырезать, Скопировать, Вставить, Отменить, Вызвать справку.** Кроме того, здесь имеется команда, вызывающая меню, элементы которого позволяют осуществить выбор активного окна **Рабочей среды**, а также команда, с помощью которой можно создавать дополнительные кнопки пользователя для быстрого запуска на выполнение наиболее часто используемых наборов команд, определенных пользователем.
2) **Набор вкладок**, позволяющий активизировать соответствующие панели инструментов.

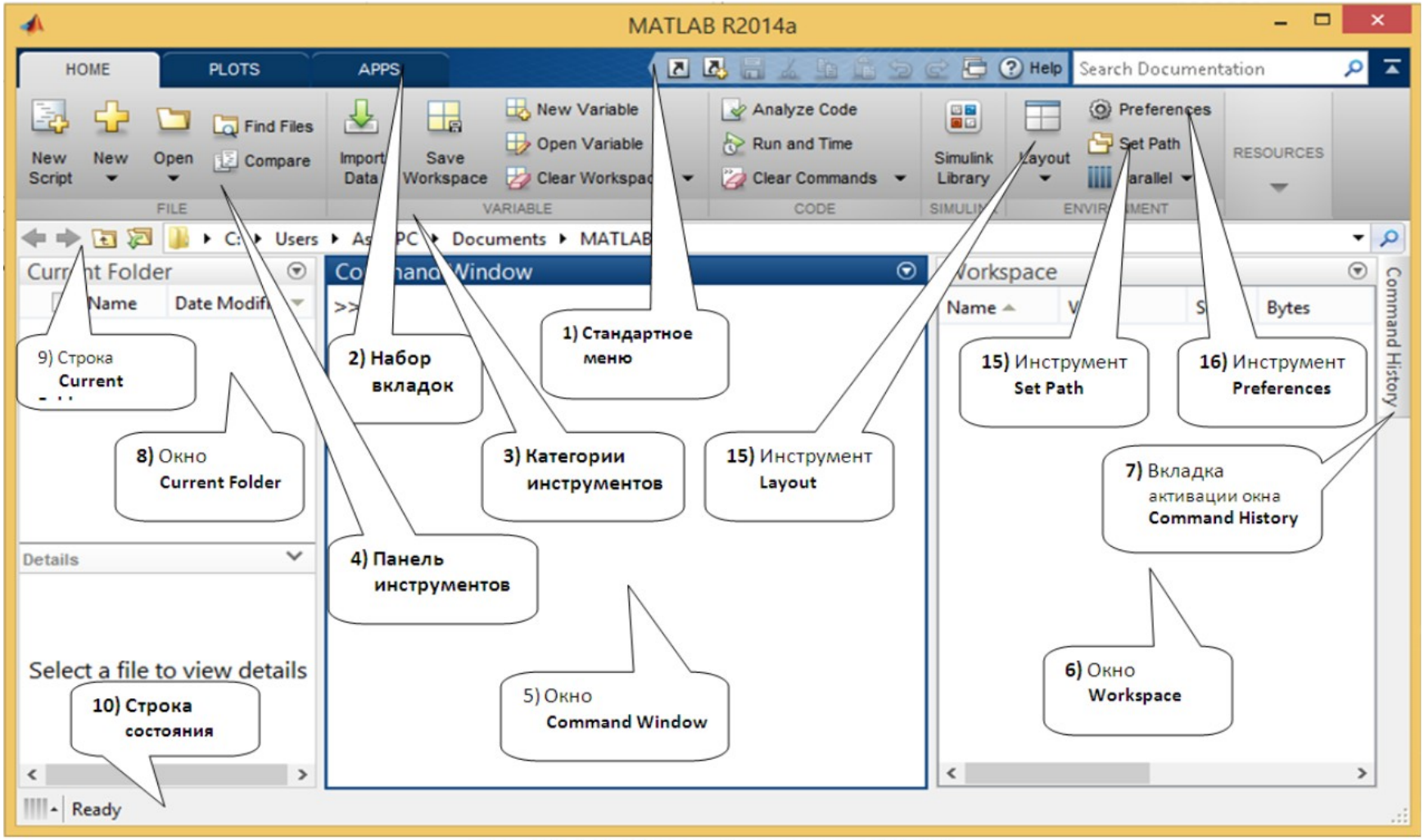


Рис. 1.1.1-1. Окно **Рабочая среда** Matlab

3) **Категории инструментов,** на которые разделен весь набор инструментов инструментальной панели соответствующих вкладок.
4) **Панели инструментов**, содержащие наборы элементов панелей инструментов, которые имеют соответствующие изображения и надписи, подсказывающие их назначение (рис. 1.1.1-1) и позволяющие выполнить наиболее часто встречающиеся команды системы Matlab. Состав набора элементов *инструментальной панели* соответствует текущей вкладке.
5) **Командное Окно** (**Command Window**) отображает вводимые команды, результаты их выполнения, а также сообщения об ошибках.
6) **Окно Рабочей Области** (**Workspace**) отображает содержимое рабочего пространства объектов Matlab, и, позволяет выполнять определенные действия с объектами этого пространства (скалярными переменными, векторами, матрицами, функциями и др.).
7) **Окно Истории Команд** (**Command History**) осуществляет просмотр и повторный вызов ранее введенных команд. Причем само окно на рис. 1.1.1-1 не показано, а отображено вкладкой для перехода в это окно.
8) **Окно текущего каталога** (**Current Folder**) предназначено для просмотра путей доступа файловой системы. В нем перед работой в Matlab с конкретным файлом (чтение или запись файла на внешний носитель), расположенным в определенной папке, необходимо указать путь доступа к файлу. В низу окна отображается информация о типе выбранного файла.
9) **Строка текущего каталога** предназначена для быстрого изменения текущего каталога.
10) **Строка состояния** отображает системные сообщения.
11) **Окно Редактор m-файлов** (**Matlab Editor**) используется для создания и отладки **m**-файлов.
12) **Окно Редактор данных** (**Array Editor**) используется для визуального просмотра и редактирования в основном для одно- или двумерных массивов, которые находятся в рабочей области.
13) **Профилировщик (Profiler)** представляет собой графический интерфейс пользователя, помогающий улучшить скорость работы m-файлов.
14) **Окно для отображения графиков (Figure)**.

Обратите внимание, что два окна Workspace и Command History могут поочередно закрывать друг друга. Для активизации нужного окна необходимо щелкнуть по соответствующей вкладке закрытой панели (рис. 1.1.1-1).

В стандартной конфигурации три основных окна, вписанные в **Рабочую среду** Matlab, закреплены (поставлены на якорь), т.е. они могут передвигаться вместе с основным окном и вместе с ним изменять свои размеры. Компоненты с 1 по 10 почти всегда присутствуют в окне **Рабочей среды (**рис. 1.1.1-1), в то время как компоненты с 11 по 14 (рис. 1.1.1-1) являются контекстно-зависимыми и появляются в **Рабочей среде** по мере активизации соответствующих инструментов.

Границы между окнами можно изменять, для этого окно нужно открепить (снять с якоря), и только тогда оно может занимать автономную позицию на экране. Расположение и границы всех окон можно настраивать командами окна **SELECT LAYOUT**, которые открываются как нажатием на инструмент **Layout**-макет рабочего окна (рис. 1.1.1-2), расположенный на инструментальной панели, так и командами ***контекстных меню*** соответствующих окон.

Контекстные меню любого окна можно отобразить на экране, щелкнув правой кнопкой мыши в область или на заголовок соответствующего окна при условии, что это окно активно. В дальнейшем первое контекстное меню будем на рисунках обозначать, как а), второе – как б).

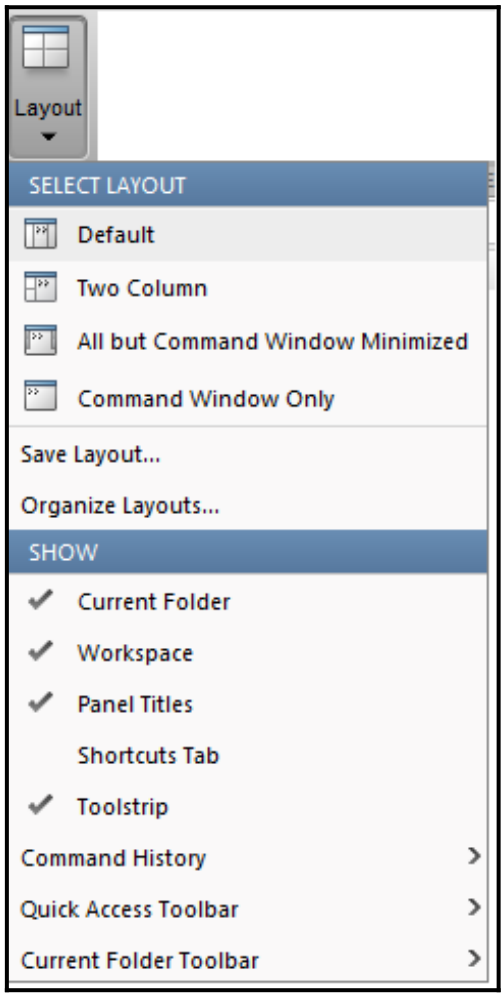


Рис. 1.1.1-2. Окно меню **SELECT LAYO**UT инструмента **Layout**

Рассмотрим основные компоненты **Рабочей среды** (рис. 1.1.1-1) в стандартной конфигурации (**Default**) более подробно.

Набор вкладок интерфейса Matlab расположен вверху **Рабочей среды,** и после загрузки Matlab состоит из трех элементов: **HOME, PLOTS, APPS**. Каждому из этих элементов соответствует своя панель инструментов.

При активизации элемента **HOME** (рис. 1.1.1-1) отображается панель инструментов, в которой присутствуют инструменты, необходимые для текущей работы с Matlab. Эти инструменты разбиты на следующие категории:

- ***FILE*** – категория, включающая инструменты, которые позволяют создавать новые наборы команд и программы и сохранять их в файлах; открывать существующие наборы команд и программ и загружать их из файлов; создавать различные объекты Matlab, осуществлять поиск файлов различных типов т.д.:
  - **New Script** – открывает редактор для написания процедур (**m**-файлов). При этом в **Рабочей среде** добавляются еще три вкладки и соответствующие им панели инструментов. Подробнее остановимся на этих инструментах в **Теме 1.3**.
  - **New** – создает новый документ Matlab. Это может быть: **m**-файл, пример, класс, системный объект, график и др.;
  - **Open** – открывает один из имеющихся документов;
  - **Find Files** и **Compare** – найти файлы и сравнить.
- ***VARIABLE*** – категория, включающая инструменты, которые позволяют осуществлять различные действия с содержимым окна **Workspace**:импорт данных, сохранение данных области **Workspace**, создание новых переменных или сделать доступными существующие данные, а также обнулять область данных:
  - **Import Data** – импортирует данные (файлы с различными расширениями) в систему Matlab;
  - **Save Workspace** – сохраняет данные области **Workspace**;
  - **New Variable** – открывает редактор для создания и редактирования переменных области **Workspace**. При этом в **Рабочей среде** добавляются еще две вкладки и соответствующие им панели инструментов. Подробнее об этих инструментах в **Теме 1.2**.
  - **Open Variable** – открывает редактор для просмотра и редактирования переменных области **Workspace.**
  - **Clear Workspace** – удаляет переменные из области **Workspace**.
- **CODE** – категория, инструменты которой позволяют организовать анализ кода и профилирование, а также обнулить содержимое окон **Command Window** и **Command History.**

- **SIMULINK** – категория, инструменты которой позволяют вызывать и работать с моделями **Simulink**;
- **ENVIROMENT** – категория, инструменты которой позволяют устанавливать свойства объектов **Рабочей среды** системы Matlab и пути доступа к файлам:
  - **Layout** – инструмент (выноска 15 рис. 1.1.1-1) открывает окно **SELECT LAYOUT**, команды которого устанавливают вид **Рабочей среды** (рис. 1.1.1-2);
  - **Preferences** – инструмент (выноска 16 рис. 1.1.1-1) открывает окно, команды которого устанавливают свойства объектов среды системы Matlab (рис. 1.1.1-3);

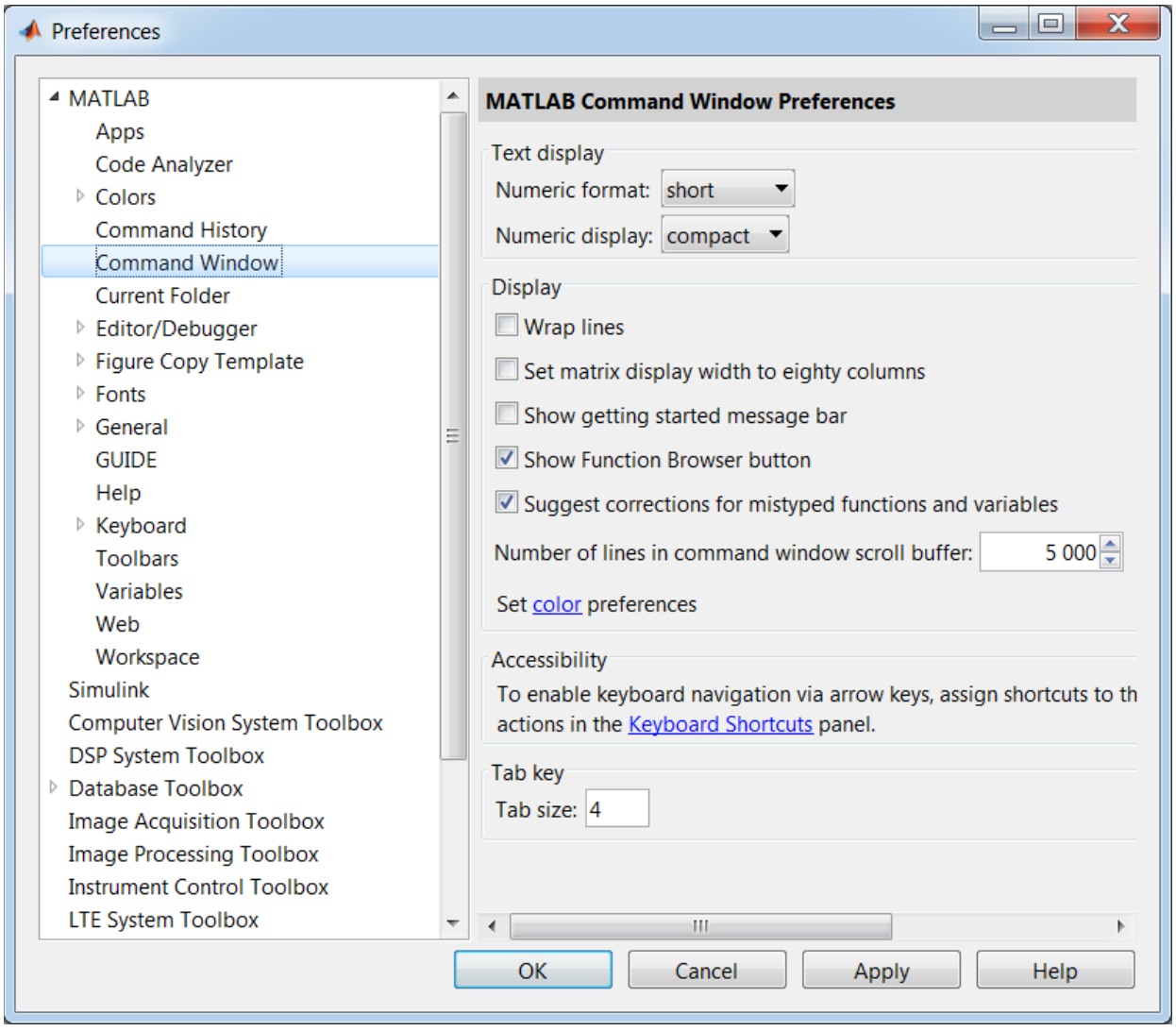


Рис. 1.1.1-3. Окно **Preferences**, в котором устанавливают свойства объектов среды системы Matlab

  - **SetPath** – инструмент (выноска 17 рис. 1.1.1-1) открывает окно установки пути для текущих файлов (рис. 1.1.1-4), которое будет использоваться при работе **Рабочей среды** Matlab. Для указания

или добавления пути к файлу, расположенному в новой папке или на новом носителе, служит редактор доступа файловой системы. Его окно открывается кнопкой инструментальной панели **Set Path** (Установить путь) (рис. 1.1.1-4). Окно показывает список папок с файлами Matlab.

Имеется возможность переноса папок вверх или вниз по списку, их уничтожения и переименования. По умолчанию задается правильная установка путей доступа.

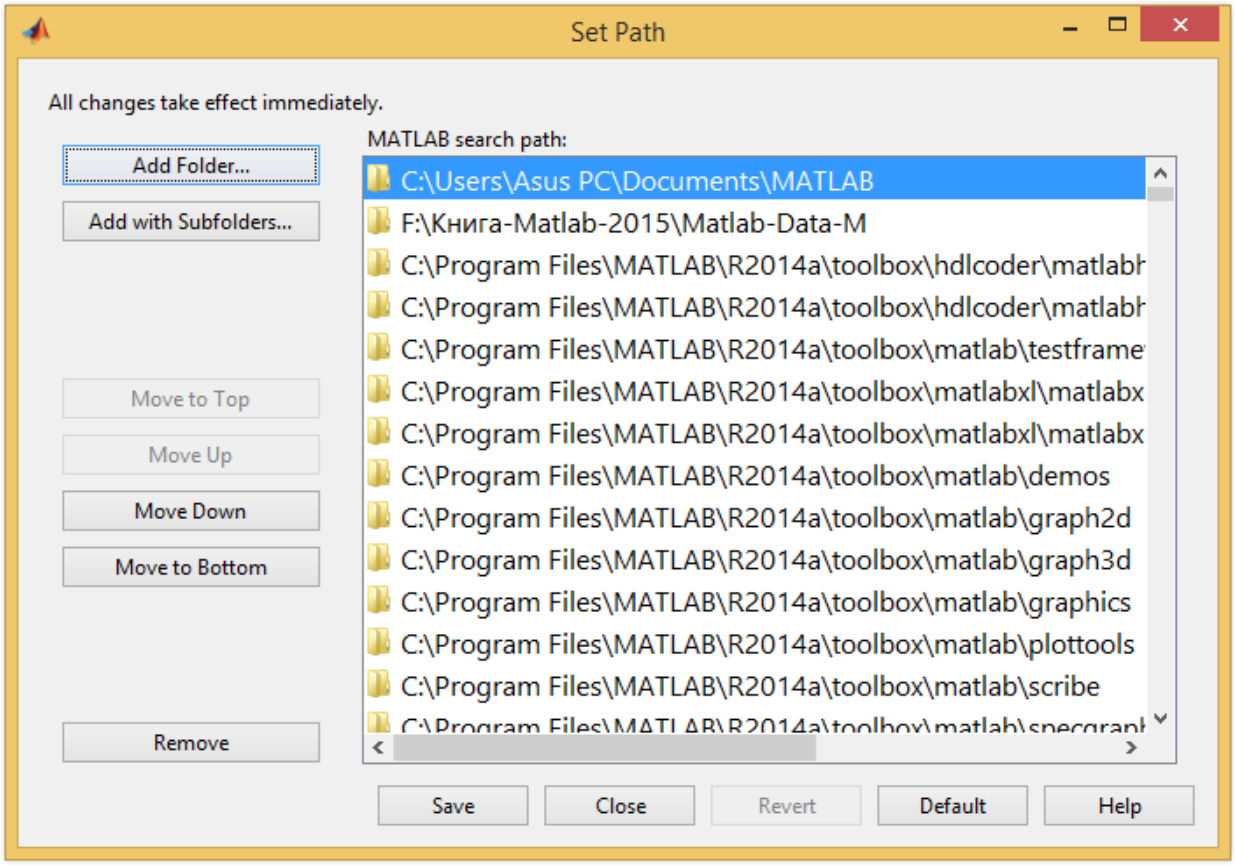


Рис. 1.1.1-4. Окно установки пути для текущих файлов

- **Parallel** – используется для параллельных вычислений.

- **RESOURSES** – категория, инструменты которой позволяют осуществлять доступ к **Справочной системе** Matlab и организовать поддержку этой системы компанией MathWorks.

При активизации элемента **PLOTS** панель инструментов приобретает вид, показанный на рис. 1.1.1-5.

Инструменты этой панели используются для работы с различными типами графиков. Например, если имеется два вектора: **M1** – значения аргументов, а **M2** – соответствующие значения функции, то после

активизации инструмента **plot** в окне **Figure 1** будет отображен соответствующий график (рис. 1.1.1-6).

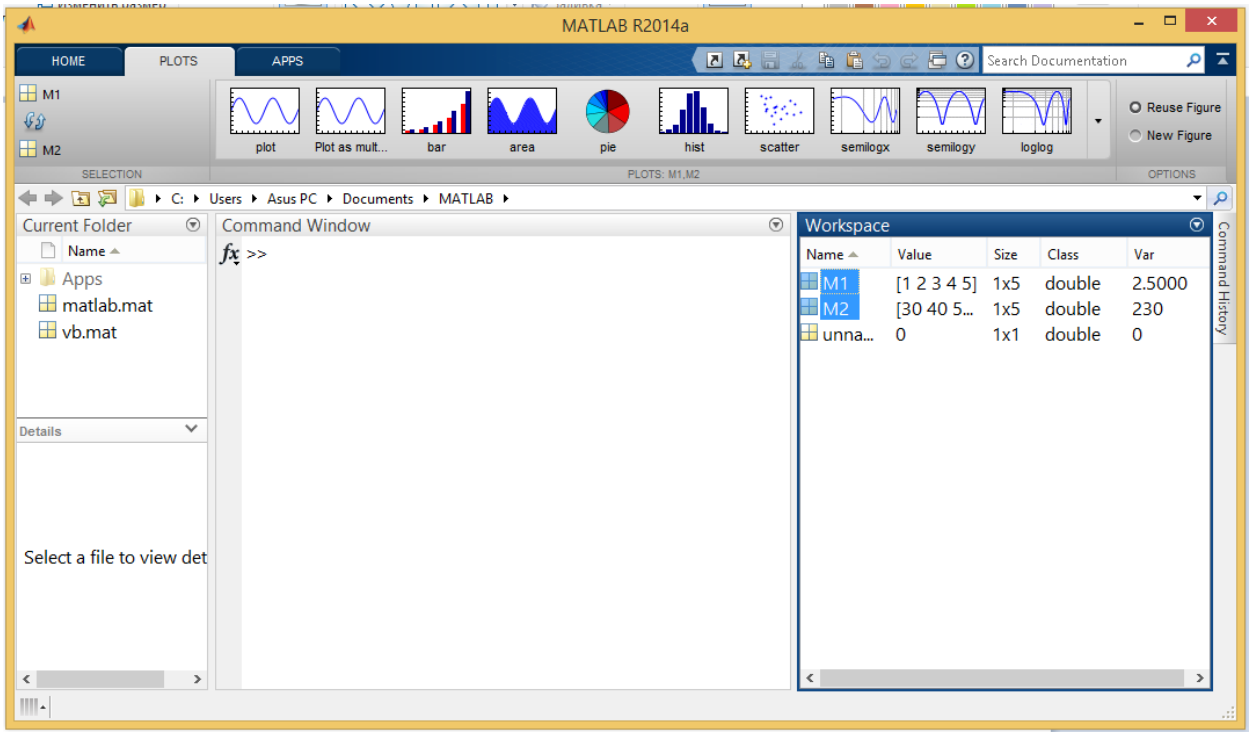


Рис. 1.1.1-5. **Рабочая среда** Matlab при активной вкладке **PLOTS**

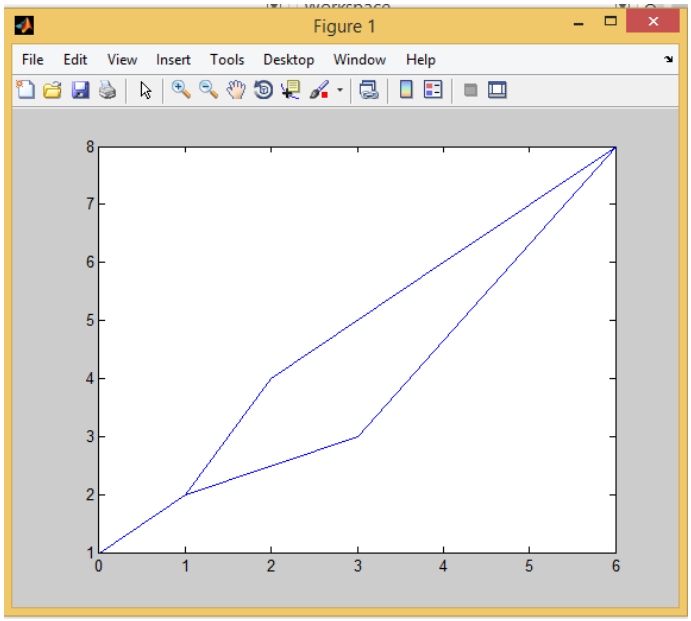


Рис. 1.1.1-6. Графики, построенные с использованием инструмента **plot**

При активизации элемента вкладки **APPS** панель инструментов приобретает вид, показанный на рис. 1.1.1-7.

Инструменты этой панели используются для работы с различными типами программных приложений и их интеграции с Matlab и Simulink.

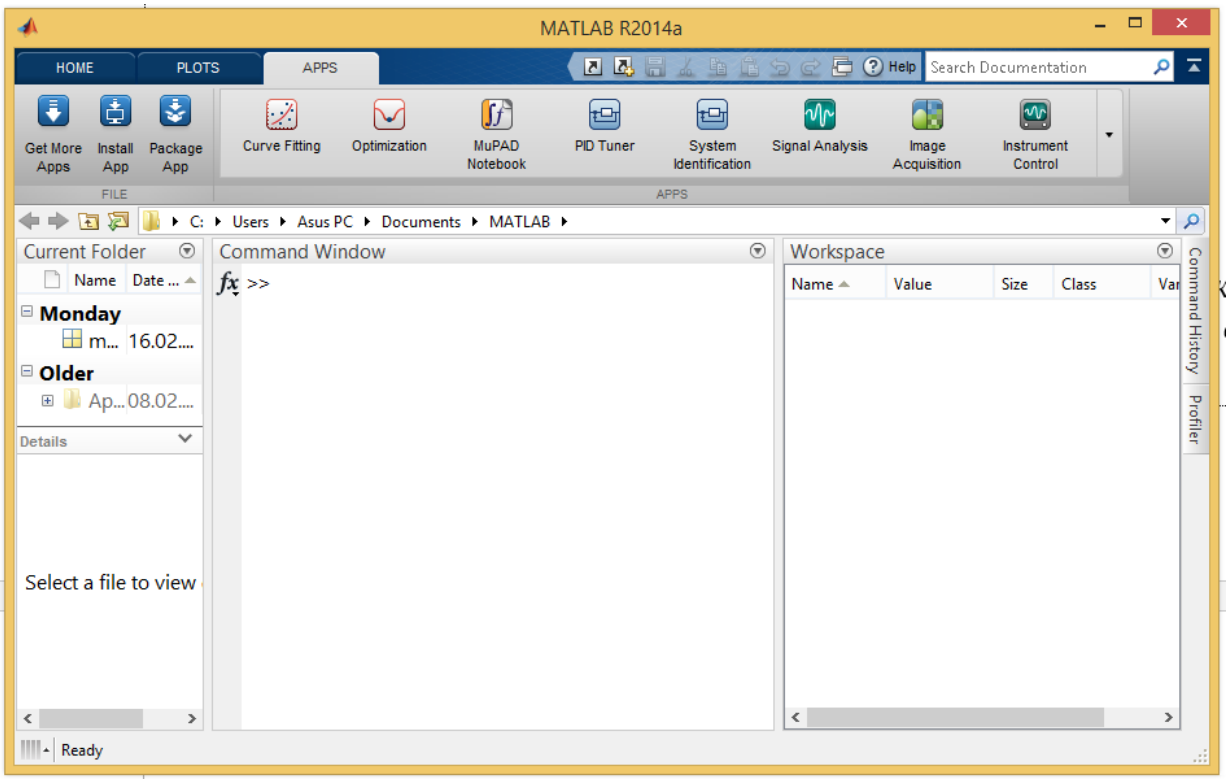


Рис. 1.1.1-7. **Рабочая среда** Matlab при активной вкладке **APPS**

Рассмотрим правила работы в окне **Command Window**.

Командное окно **Command Window** используется для ввода команд с соответствующими данными и вывода результатов их выполнения. Работа происходит в диалоговом режиме: пользователь вводит команду и передает ее ядру Matlab, ядро обрабатывает полученную команду и возвращает результат. Все команды вводятся в командную строку после появления ***приглашения >>***. *Заканчивается ввод каждой командной строки нажатием клавиши* <Enter>.

Вышеописанный сеанс работы с Matlab в окне **Command Window** принято называть *сессией*.

Рассмотрим несколько простых примеров, выполненных в командном окне (рис. 1.1.1-8).

В Примере 1 вычисляется результат выражения **2+3**. Matlab по умолчанию создаёт переменную с именем **ans**, в которую записывает значение результата текущей операции и выводит его в следующей строке.

В Примере 2 создаётся переменная **a**, вычисляется значение выражения **sin(pi),** и результат присваивается переменной **a**. Теперь эта переменная определена (атрибуты этой переменной отображаются в окне **Workspace**) и ее можно использовать для дальнейших вычислений.

В Примере 3 выражение заканчивает точка с запятой, которая «гасит» вывод результата, но он по-прежнему сохраняется, в этом случае в переменной с именем **r**, при этом значение этой переменной также отображается в окне **Workspace**. Значение этой переменной можно вывести в

строке окна **Command Window** в любой момент, набрав имя переменной в командной строке и нажав клавишу <Enter>.

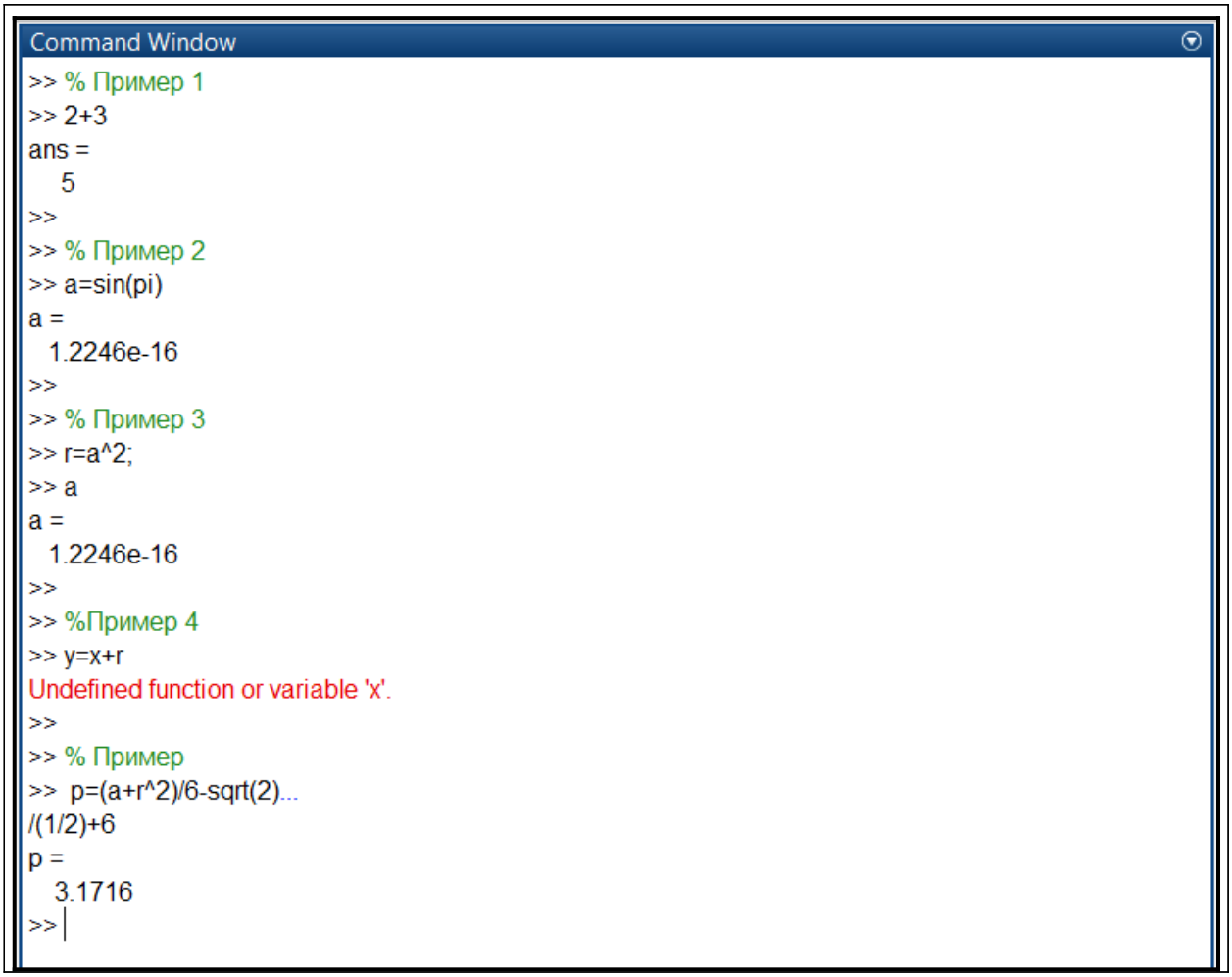


Рис. 1.1.1-8. Примеры простейших вычислений в окне **Command Window**

Обратите внимание, что предпочтительнее вычислять длинное выражение по частям с использованием промежуточных переменных.

Обратите внимание, что в любой момент значение переменной можно отобразить в командном окне, набрав имя переменной и нажав <Enter>, либо использовать функцию disp(), например, disp(b).

Все переменные системы Matlab размещаются в рабочей области оперативной памяти и отображаются в окне **Workspace** (рис. 1.1.1-9). Информацию о них (имена, размерности, типы и др.) можно отобразить на экране с помощью команд Matlab **who** или **whos**.

Для очистки командного окна служит команда **clc**, а для очистки рабочей памяти – **clear all**.

Обратите внимание, что переменные объявлять не надо, так как по умолчанию все вычисления в Matlab выполняются с двойной точностью.

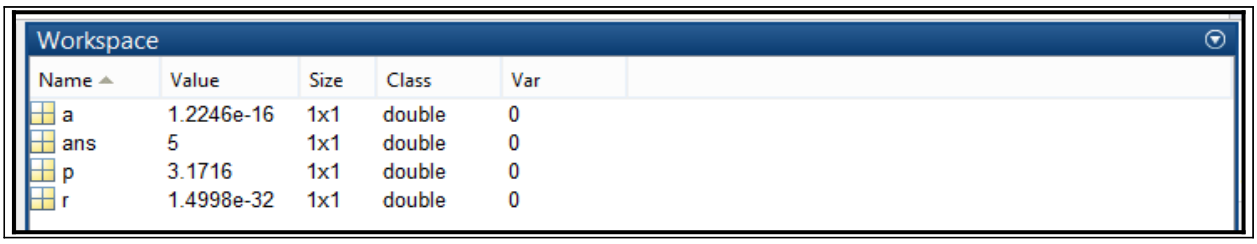


Рис. 1.1.1-9. Окно рабочей области Workspace

Формат вывода численных значений на экран можно установить принудительно в окне **Command Window**, указав тип формата командой:

format *ТипФормата*

или с помощью установки свойств в окне **Preferences**, которое можно вызвать соответствующим инструментом панели инструментов вкладки **Home**
(рис. 1.1.1-1).

Обратите внимание, что в окне Preferences можно установить свойства почти всех объектов Рабочей среды Matlab.

Для установки формата в окне **Preferences** внутри группы **Text display** следует выбрать раскрывающиеся списки **Numeric format**, а из раскрывшегося списка **Numeric format** выбрать один из форматов табл. 1.1.1-1.

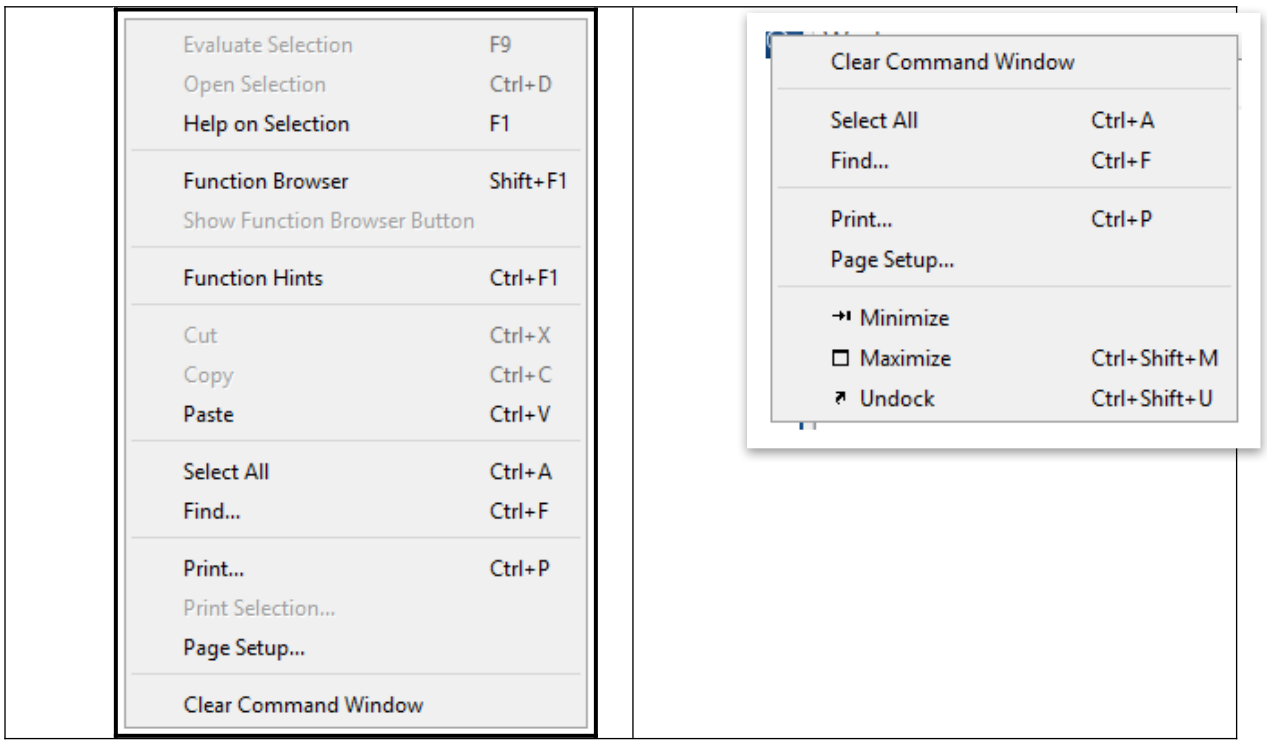
Таблица 1.1.1-1

Формат	**Пояснения**
short	Выводит короткое число с плавающей точкой. Число представляется с помощью четырех цифр после десятичной точки (по умолчанию)
long	Выводит длинное число с плавающей точкой. Представляется с помощью четырнадцати цифр после десятичной точки
short e	Выводит короткое число с плавающей точкой. Представляется с помощью пяти разрядов, четыре из которых отводится под вывод дробной части
long e	Выводит длинное число с плавающей точкой. Представляется с помощью шестнадцати разрядов, пятнадцать из которых отводится под вывод дробной части

В стандартной конфигурации **Рабочей среды** для выделения результатов вычисления или значений переменных Matlab перед выводимым значением вставляет пустую строку. Управлять появлением или отсутствием пустой строки можно в диалоговом окне **Preferences** внутри группы **Text display** с помощью форматов:

- **compact** – строки с результатами выводятся подряд;
- **loose** – строки с результатами разделяются пустой строкой.

Если окно **Command Window** активно (заголовок окна выделен синим цветом), то после нажатия на правую кнопку мышки, указатель которой находится на заголовке окна, появится контекстное меню, показанное на рис. 1.1.1-10а, а если указатель мышки находится в области окна, появится контекстное меню, показанное на рис. 1.1.1-10б.



а) **б)**

Рис. 1.1.1-10. Контекстные меню Command Window

Сохранить содержимое рабочего окна (сессию) на внешнем носителе в виде текстового файла можно с помощью специальных команд для ведения, так называемого дневника сессии:

diary *ИмяФайла***.m** или **diary** *ИмяФайла***.txt.**

Эти команды производят запись в текстовый файл всех команд ввода с клавиатуры и полученных результатов. Чтобы записать части содержимого **Командного окна**, используются команды:

diary off – приостанавливает запись в файл, а команда

diary on – вновь начинает запись в файл.

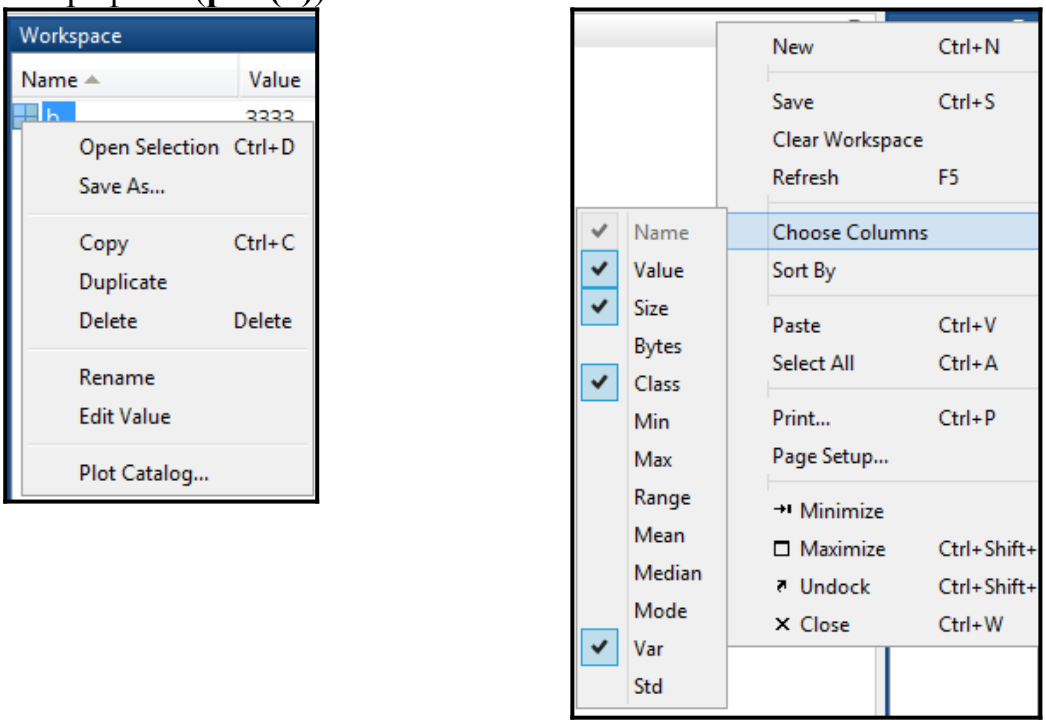
Таким образом, чередуя команды **diary off** и **diary on**, можно сохранять необходимые фрагменты сессии.

Окно **Workspace** предназначено для быстрого просмотра атрибутов переменных, расположенных в рабочей области, а также их записи в файл и чтения из файла (рис. 1.1.1-1). В этом окне можно увидеть имя переменной (**Name**), значение (**Value**), ее размер (**Size**), число байтов, занимаемых

переменной в памяти (**Bytes**), и ее тип (**Class**). Для идентификации класса переменной слева от имени используется соответствующая иконка.

Отобразить на экране окно **Workspace** можно либо с помощью соответствующей команды инструментальной панели, либо с помощью задания команды **Workspace** в командном окне. С помощью команд контекстного меню (рис. 1.1.1-11а,б) можно изменять внешний вид окна **Workspace** (скрывать или показывать поля **Size, Value, Bytes** и **Class**), а также сортировать переменные по имени, размерности, количеству байт и классу.

Команды контекстного меню**Workspace** позволяют выполнять следующие действия: создать новую переменную (**New**); открыть редактор данных для просмотра или редактирования значений выделенной переменной (**Open Selection**); загружать данные из файла в рабочую область (**Load**); сохранять рабочую область в файле (**Save**); печатать содержимое рабочей области (**Print**); удалить выделенную переменную (**Delete**); построить графики (**plot(d)**).



а) **б)**

Рис. 1.1.1-11. Контекстные меню окна **Workspace**

Просмотреть содержимое рабочей области, загружать или удалять данные можно и с помощью ввода команд в окне **Command Window**. Так для просмотра переменных, находящихся в рабочей области, в командную строку необходимо ввести команду **who**. Результат выполнения команды **who** по внешнему виду напоминает окно **Workspace**, только без полей **Size, Bytes** и **Class**.

Полную информацию о содержании рабочей области можно получить с помощью команды **whos**, в результате выполнения которой в окне **Com-**

mand Window выводится информация обо всех переменных и общий объем занимаемой ими памяти.

Для удаления переменной из рабочей области следует ввести команду

clear *ИмяПеременной.*

Очистка всей рабочей области осуществляется с помощью команды

clear без параметров.

Загрузка всех данных из файла реализуется командой

load *ИмяФайла.*

Выборочная загрузка данных реализуется командой

load *ИмяФайла ИмяПеременной.*

Для сохранения рабочей области на диске необходимо ввести команду

save *ИмяФайла.*

Данные будут сохранены в файле с расширением ***.mat**. Выборочное сохранение переменных из рабочей области обеспечивается командой

save *ИмяФайла ИмяПеременной.*

Окно **Редактор данных** изображено на рис 1.1.1-12, предназначено для просмотра и редактирования значений переменных. Под редактированием переменных подразумевается не только изменение значений элементов массива, но также и изменение размера массива.

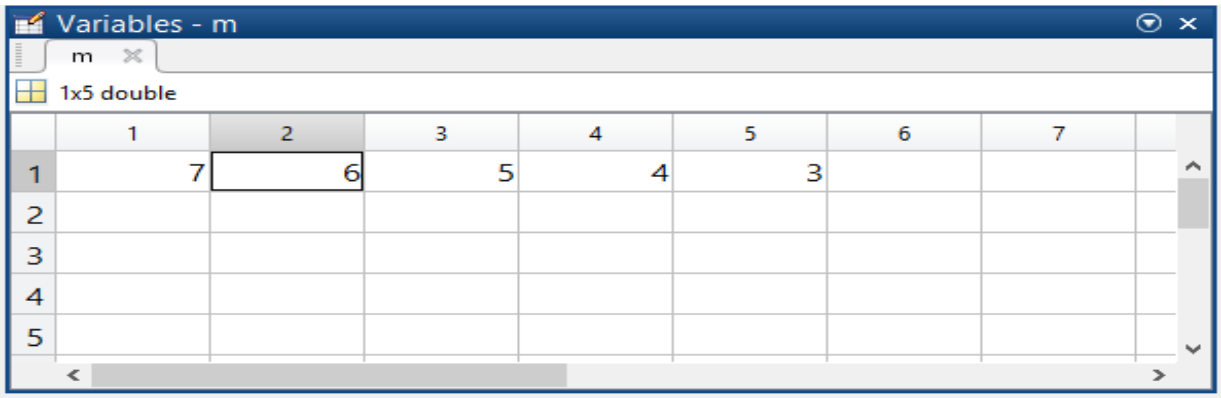
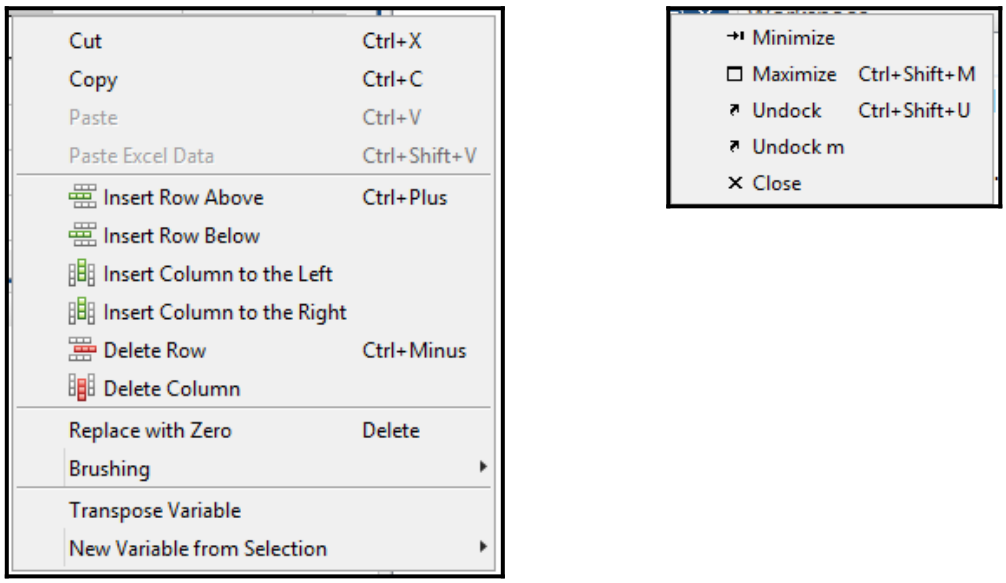


Рис. 1.1.1-12. Окно редактора данных



а) **б)**

Рис. 1.1.1-13. Контекстные меню окна **Редактор данных**

Редактор данных вызывается двойным щелчком на имени переменной в окне рабочей области или заданием в командном окне команды **openvar('***ИмяПеременной***')**.

Контекстные меню **Редактор данных** показаны на рис. 1.1.1-13.

На рис. 1.1.1-14 показан пример окна **Command History.** Это окно служит для просмотра команд, заданных ранее в командной строке **Command Window.**

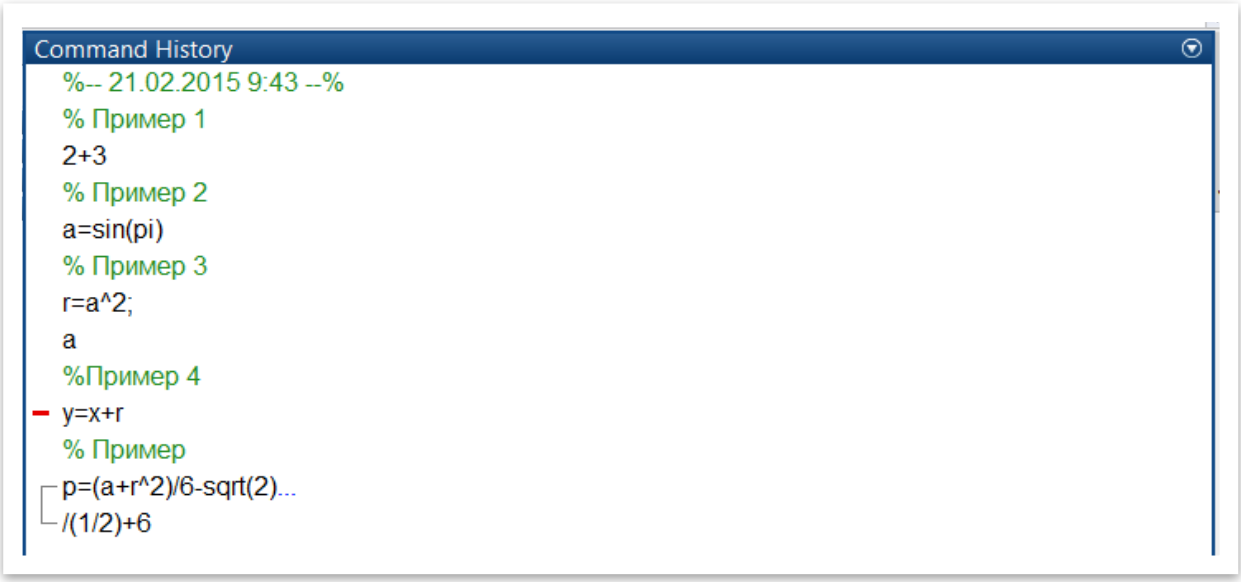
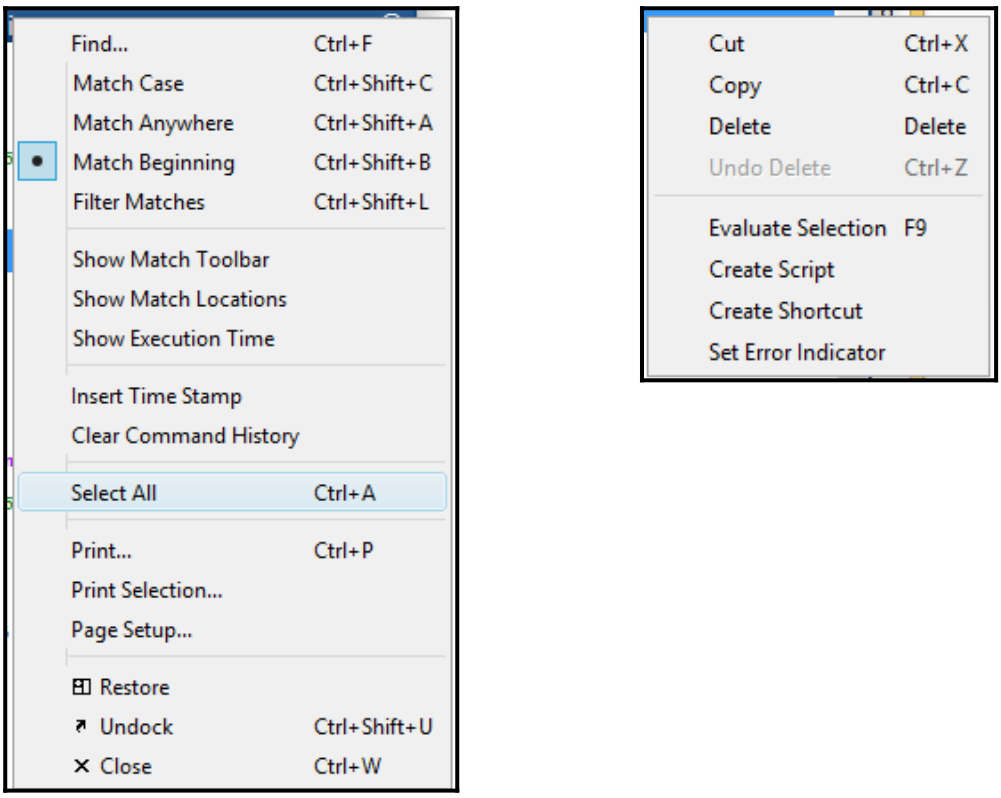


Рис. 1.1.1-14. Окно истории команд **Command History**



а) **б)**

Рис. 1.1.1-15. Контекстное меню окна **Command History**

Изменять параметры работы окна истории команд можно с помощью диалогового окна **Preferences** (рис. 1.1.1-3), при активном окне **Comman History**. Окно **CommandHistory** хранит все команды, набираемые пользователем. В отличие от содержимого **Command Window** сюда не попадают сообщения системы и результаты вычислений.

Окно **Command History** целесообразно использовать при вводе команд, однотипных вычислений или просто повторений группы командных строк. Чтобы ввести в текущую строку содержимое ранее введенной командной строки, достаточно нажатием клавиш <↑> или <↓> подобрать нужную строку. Перенести конкретную строку в командное окно можно двойным щелчком мыши по этой строке в окне **Command History**.

Для очистки области истории команд используются команда **clear** или команда контекстного меню **clear Command History.**

1.1.2. Основные объекты системы Matlab

Математические выражения в Matlab строятся на основе чисел, числовых констант, переменных, стандартных и нестандартных функций, соединенных знаками арифметических операций и круглых скобок. Кроме того, в математических выражениях могут использоваться различные спецзнаки. Вид результата зависит от установленного формата.

Число – объект языка Matlab**,** представляющий числовые данные. Числа могут быть представлены в целом, дробном, с фиксированной и плавающей точкой, а также в экспоненциальном виде. Например,

0, 2, -4, 4.67, 0.0005, 567.9e-7, 0.89e12.

Причем, числа могут быть как **действительными**, так и **комплексными.** Комплексные числа содержат действительные и мнимые части. В Matlab мнимая часть имеет множитель **i** или **j,** означающий корень квадратный из **-1.** Например,

3i, 5j, -5.1 + i8, 0.05e-5 - j0.006.

Числовая константа – это предварительно определенное число (числовое значение). Числа (например, **1, -5, 3.97**) являются безымянными числовыми константами.

Системные константы (табл. 1.1.2-1) – это такие константы, значения которых задаются системой при загрузке, но при необходимости их можно переопределить.

Таблица 1.1.2-1

Константа	Назначение
i или **j**	Мнимая единица
pi	Число **π=3.1415926…**
eps	Погрешность вычислений над числами с плавающей точкой (**2^{-52}**)
realmin	Наименьшее число с плавающей точкой (**2^{-1022}**)
realmax	Наибольшее число с плавающей точкой (**2^{1022}**)
inf	Значение машинной бесконечности
ans	Переменная, хранящая результат последней операции
NaN	Указание на нечисловой характер данных **(Not-a-Number)**

Переменные – это объекты, имеющие имена. Они способны хранить некоторые разные по значению данные. В зависимости от этих данных переменные могут быть числовыми или символьными, векторными или матричными. Имена переменных (идентификаторы) задаются по следующим правилам:

- имя переменной может состоять из символов латинского алфавита, знака подчёркивания и цифр, но начинаться обязательно с буквы;
- в имени переменной различаются прописные и строчные буквы;
- в имя переменной не входит пробел;

- имя переменной не должно совпадать с именами других переменных, функций, то есть должно быть уникальным;
- желательно использовать содержательные имена для обозначения переменных.

Обратите внимание, что переменные и их типы заранее не декларируются (не объявляются). Они объявляются выражением, значение которого после вычисления присваивается переменной.

В оперативной памяти компьютера переменные занимают определенное количество байтов. Как известно, эта область называется **Workspace.**

Операция присваивания используется для задания переменным определенных значений и обозначается знаком равенства =:

Имя переменной = Выражение

Имя переменной = Выражение;

Арифметические операции в системе Matlab можно проводить вычисления как с вещественными, так и с комплексными числами. Полный список арифметические операции приведен в табл. 1.1.2-2.

Следует обратить внимание, что в математических выражениях с операциями +, -, * ,^, /, \ участвуют как векторы, так и матрицы.

Первые восемь операций действуют в соответствии с правилами линейной алгебры, т.е. с правилами операций над векторами и матрицами.

Последние четыре операции, так называемые операции с точкой, осуществляют поэлементные операции над массивами.

Таблица 1.1.2-2

Функция	Название	Операция	Синтаксис
plus()	Плюс	+	**M1+M2**
uplus()	Унарный плюс	+	**+M**
minus()	Минус	-	**M1-M2**
uminus()	Унарный минус	-	**-M**
mtimes()	Матричное умножение	*	**M1*M2**
times()	Поэлементное умножение массивов	.*	**A1.*A2**
mpower()	Возведение матрицы в степень	^	**M^x**
power()	Поэлементное возведение массива в степень	.^	**A.^x**
mldivide()	Обратное (справа налево) деление матриц	\	**M1\M2**
mrdivide()	Деление матриц слева направо	/	**M1/M2**
ldivid()	Поэлементное деление массивов	.\	**A1.\A2**

	справа налево		
rdivide()	Поэлементное деление массивов слева направо	**./**	**A1./A2**

Подробнее операции над векторами и матрицами рассмотрены в **Теме 1.2**.

Приоритет исполнения математических операций в Matlab аналогичен приоритету в языках программирования высокого уровня: возведения в степень, умножения и деления, сложения и вычитания. Для изменения приоритета в математических выражениях используются круглые скобки.

Функции – это имеющие уникальные имена объекты, выполняющие определенные преобразования своих аргументов (параметров) и при этом возвращающие результаты этих преобразований. **Возврат результата** – отличительная черта функций. Функции могут быть ***встроенными*** (внутренними) ***функциями*** системы Matlab***, функциями пользователя***, которые могут дополнять встроенные функции и описанные конкретным пользователем для своих нужд, ***внешними функциями***, или ***m-функциями***.

Matlab обладает большим набором элементарных ***встроенных (библиотечных) функций,*** в том числе математических**.** Встроенные функции хранятся в откомпилированном виде в **ядре** системы Matlab. Некоторые из них приведены в табл. 1.1.2-3.

Таблица 1.1.2-3

Тригонометрические функции (аргумент задается в радианах)	
sin(), cos(), tan(), cot()	Синус, косинус, тангенс и котангенс
sec(), csc()	Секанс, косеканс
Обратные тригонометрические функции (результат в радианах)	
asin(),cos(),atan(), atan2(), acot()	Арксинус, арккосинус, арктангенс и арккотангенс
asec(), acsc()	Арксеканс, арккосеканс
Алгебраические и арифметические функции	
abs(), sign()	Модуль и знак числа
exp()	Экспоненциальная функция
log(), log2(), log10()	Логарифм натуральный, по основанию 2 и 10
sqrt()	Квадратный корень
fix()	Целая часть числа
floor()	Округление до ближайшего целого значения, которое не превышает аргумент
mod(x, y), rem(x, y)	Остаток от деления x на y. Целая часть определяется соответственно функциями floor и fix
sign()	Знак числа
factorial()	Вычисление факториала числа
Гиперболические функции	
sinh(), cosh(), tanh(), coth()	Гиперболические синус, косинус, тангенс и котангенс

sech(), csch()	Гиперболические секанс и косеканс
asinh(), acosh(), atanh(), acoth()	Гиперболические арксинус, арккосинус, арктангенс и арккотангенс
Функции для работы с комплексными числами	
conj()	Комплексно-сопряжённое число
imag()	Мнимая часть числа
real()	Вещественная часть числа
mod(), rem()	Остаток от деления с учетом знака делимого и без
gcd(), lcm()	Наибольший и наименьший общий делитель
angle()	P = angle(Z) возвращает фазу угла в радианах. Для комплексного Z модуль R и фаза угла P связаны следующим образом: R = abs(Z) theta = angle(Z) Z = R.*exp(i*theta)
isreal()	Возвращает логическую 1, если число действительное, и 0 – если комплексное.

Система Matlab содержит более 1000 встроенных функций, однако предоставляет пользователю средства для создания и использования своих собственных функций – так называемых ***функций пользователя***. О создании и использовании функций в виде **m-файлов** речь пойдет в параграфе 1.1.3.

Рассмотрим несколько примеров (рис. 1.1.2-1).

```
Command Window
>> x=10;
>> sqrt(x) %квадратный корень x
ans =
    3.1623
>> abs(-1.1) %абсолютная величина числа -1.1
ans =
    1.1000
>> sin(x)    %Sin(10)
ans =
   -0.5440
>> factorial(3) %факториал 3
ans =
     6
>> mod(7,5) %остаток от деления по модулю 7 5
ans =
     2
>> imag(1+1.5i)  %мнимая часть комплексного числа
ans =
    1.5000
```

Рис. 1.1.2-1. Примеры использования встроенных функций Matlab

Кроме математических выражений в Matlab используются *логические выражения.*

Логические выражения в Matlab строятся на основе математических выражений, операций отношения и соответствующих им функций, логических операций и функций, а также круглых скобок.

Результатом логического выражения является значение равное 1, если выражение Истинно (True), и значение 0 в противном случае – выражение Ложно (False). Если операнды – действительные числа, то использование операций тривиально.

Операции отношения и соответствующие им функции служат для сравнения двух величин, векторов и матриц. Список операций отношений приведен в табл. 1.1.2-4.

Таблица 1.1.2-4

Функция	Операция	Описание	Пример
eg()	==	Равно	x==y; eg(x,y);
ne()	~=	Не равно	x~=y; ne(x,y);
lt()	<	Меньше чем	x<y; lt(x, y);
gt()	>	Больше чем	x>y; gt(x,y);
le()	<=	Меньше или равно	x<=y; le|(x,y);
ge()	>=	Больше или равно	x>=y; ge(x,y);

Операции отношений выполняют поэлементное сравнение векторов или матриц одинакового размера и возвращают значение равное 1, если элементы идентичны, и значение 0 в противном случае.

Следует отметить, что операции <, <=, >, >= при комплексных операндах используются для сравнения только действительных частей операндов – мнимые отбрасываются. В то же время операции == и ~= ведут сравнения с учетом как действительной, так и мнимой частей операндов (рис. 1.1.2-2).

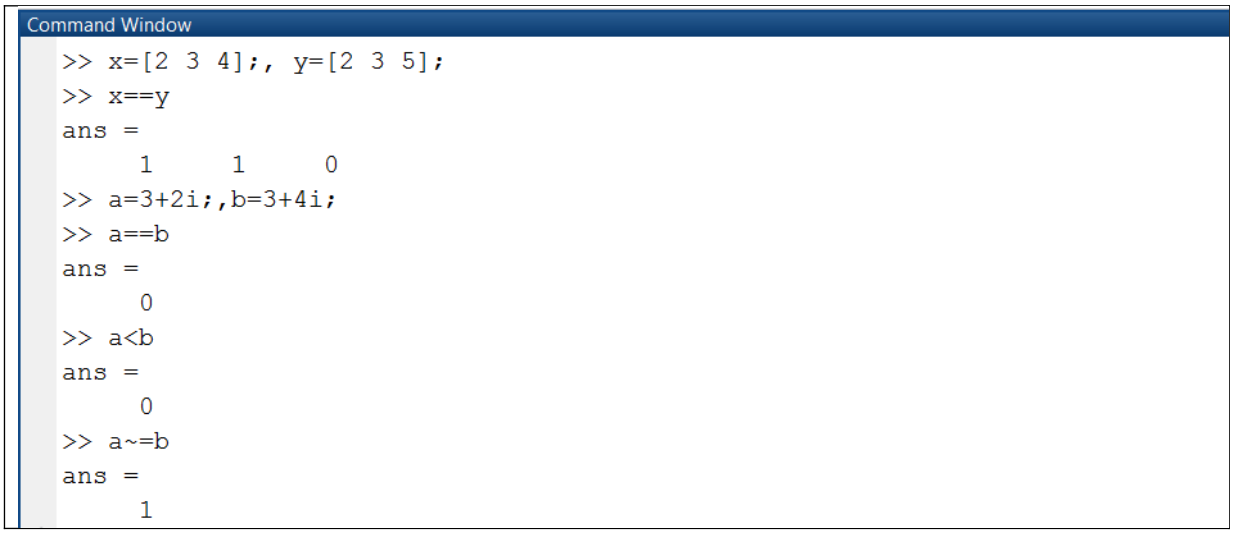


Рис. 1.1.2-2. Примеры операций отношения над векторами и комплексными числами

Операции отношения обычно применяются в логических выражениях, которые являются условиями операторов **if, for, while, switch** и служат для изменения последовательности выполнения операторов

программы Matlab.Использование этих операторов подробно рассмотрено в **Теме 1.3**.

При вычислении выражений операции отношений имеют более низкий приоритет, чем арифметические, но более высокий, чем логические операции.

Логические операции и соответствующие им функции служат для поэлементных логических операций над элементами одинаковых по размеру массивов. Список логических операций отношений приведен в табл. 1.1.2-5.

Таблица 1.1.2-5

Функция	Операция	Описание	Пример
and()	&	Логическое умножение (И)	>>x&y ans = 1 0 0
or()	!	Логическое сложение (ИЛИ)	>>or(x,y) ans = 1 1 1
not()	~	Логическое НЕ	>> ~x ans = 0 1 1

Логические функции дополняют логические операции и представлены в табл. 1.1.2-6, где в примерах используются вектора x=[1 0 0] и y=[111].

Таблица 1.1.2-6

Функция	Описание	Пример
xor ()	Исключающее ИЛИ	>>xor(x,y) ans = 0 1 1
all()	Верно, если все элементы вектора не равны 0	>>all(x) ans = 0
any()	Верно, если все элементы вектора равны 0	>>any(x) ans = 1
find()	Нахождение ненулевых элементов в векторах	>>find(x) ans = 1

Специальные операции реализуются с помощью специальных символов (табл. 1.1.2-7). Они предназначены для создания самых разнообразных объектов входного языка системы Matlab и придания им различных форм.

Таблица 1.1.2-7

Обозначение	Название

:	**Двоеточие** – формирование подвекторов и подматриц
( )	**Круглые скобки** используются для задания порядка выполнения операций в выражениях, указания последовательности аргументов функции и указания индексов элемента вектора или матрицы
[ ]	**Квадратные скобки** – формирование векторов и матриц
{ }	**Фигурные скобки** – формирование массивов ячеек
.	**Десятичная точка** используется для отделения дробной части чисел от целой
..	**Две точки подряд** указывают на родительский каталог – переход по дереву каталогов на один уровень вверх
...	**Три точки подряд** указывают на продолжение строки
,	**Запятая** – разделитель элементов
;	**Точка с запятой** используется внутри круглых скобок для разделения строк матриц, а также в конце операторов для запрета вывода на экран результата вычислений
%	**Знак процента** – начало комментария
=	**Символ равно** используется для присваивания (операция присваивания) значений в выражениях
'*текст***'**	**Одиночные кавычки (апострофы), внутри которых заключен текст,** интерпретируемый как вектор символов с компонентами, являющимися символами. Кавычка внутри строки задается двумя кавычками
'	**Кавычка (апостроф)** – транспонирование матриц (A'). Для комплексных матриц транспонирование дополняется комплексным сопряжением, т.е. строки транспонированной матрицы соответствуют столбцам исходной матрицы
.'	**Точка с кавычкой (точка с апострофом)** – транспонирование массива (A.') Для комплексных массивов операция сопряжения не выполняется
[,]	**Квадратные скобки с перечислением внутри их элементов через запятую** – горизонтальная конкатенация – объединение матриц A и B[A,B]
[;]	**Квадратные скобки с перечислением внутри их элементов через точку с запятой** – вертикальная конкатенация – объединение матриц A и B[A;B]

Функции пользователя – это функции, создаваемые пользователем системы, которые значительно облегчают работу за счет частого использования в математических выражениях, получении таблиц значений функций и построении графиков.

Как правило, функции создаются для обработки множества значений аргументов, поэтому при описании их математических выражений используются операции с точкой. Для создания функции с помощью оператора **@** используется следующий формат:

Имя **=** **@(***Аргументы***)***Выражение*

Функция может иметь несколько аргументов, тогда они перечисляются через запятую. Имя функции формируется аналогично имени переменной и

должно быть уникально. Если в конце строки не введен символ точки с запятой, то в следующей строке выводится выражение функции.

В примере (рис. 1.1.2-3) описана функция $y(x)=e^{x^2}$ и получены ее значения для множества значений аргумента **x**. Традиционное обращение к функции выводит значения функции в строку, а добавление символа апостроф (') – в столбец. Также в примере показаны возможности вычисления значения функции от числового аргумента и использования функции пользователя в выражениях.

Для создания функции с использование **inline()** применяется следующий формат:

Имя = **inline**('*Выражение*')

Здесь выражение заключено в апострофы, а аргумент функции **inline()** символьный. Функция сама распознает аргументы и выводит в следующей строке принятое обращение.

В приведенном ниже примере (рис. 1.1.2-4) показано описание функции с использованием **inline()**, вычисление значения функции **s(x,y)** для аргументов, заданных числами, и для аргументов, являющихся векторами. В последнем случае векторы, являющиеся аргументами функции, должны быть одной длины.

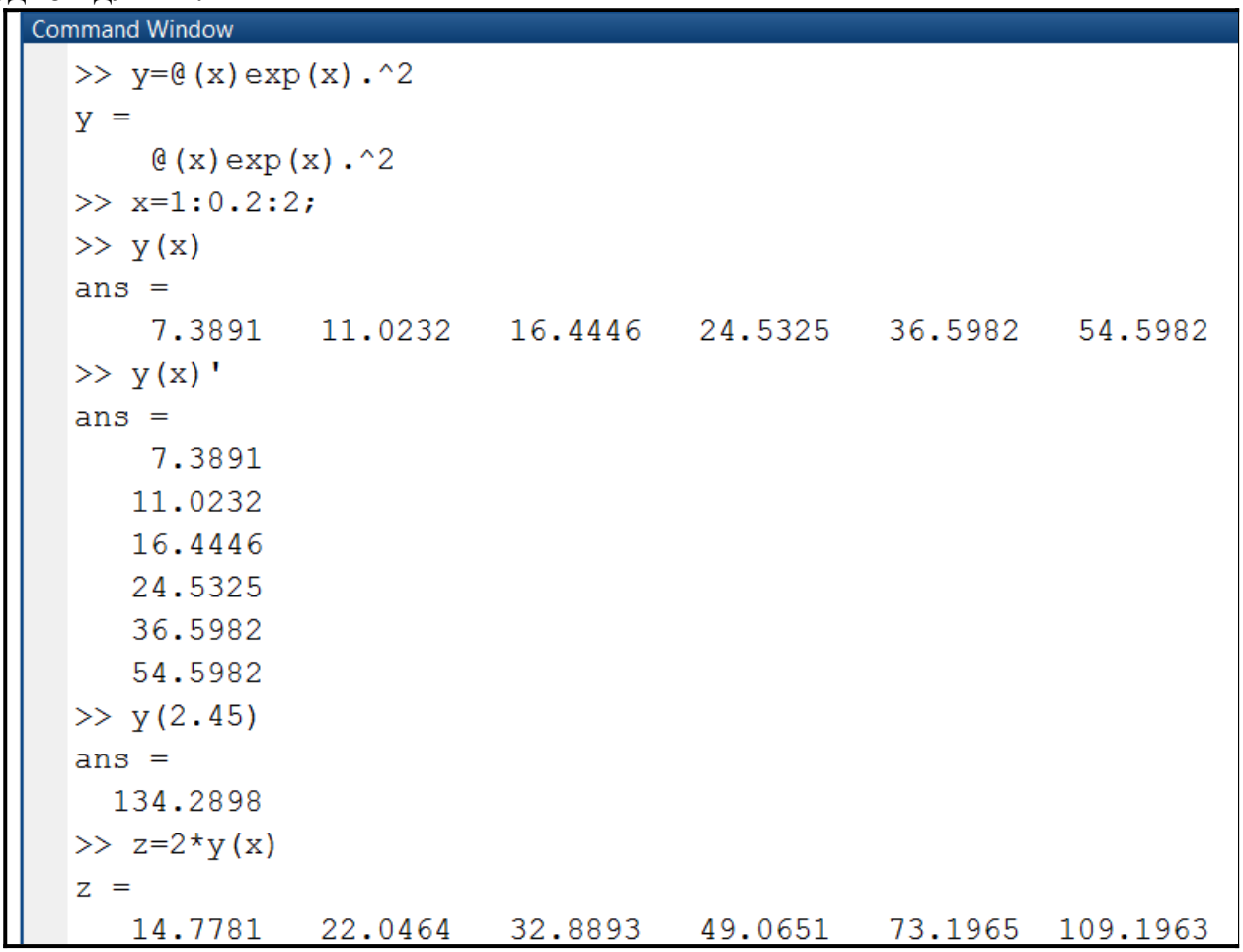


Рис. 1.1.2-3. Создание функции с помощью оператора @

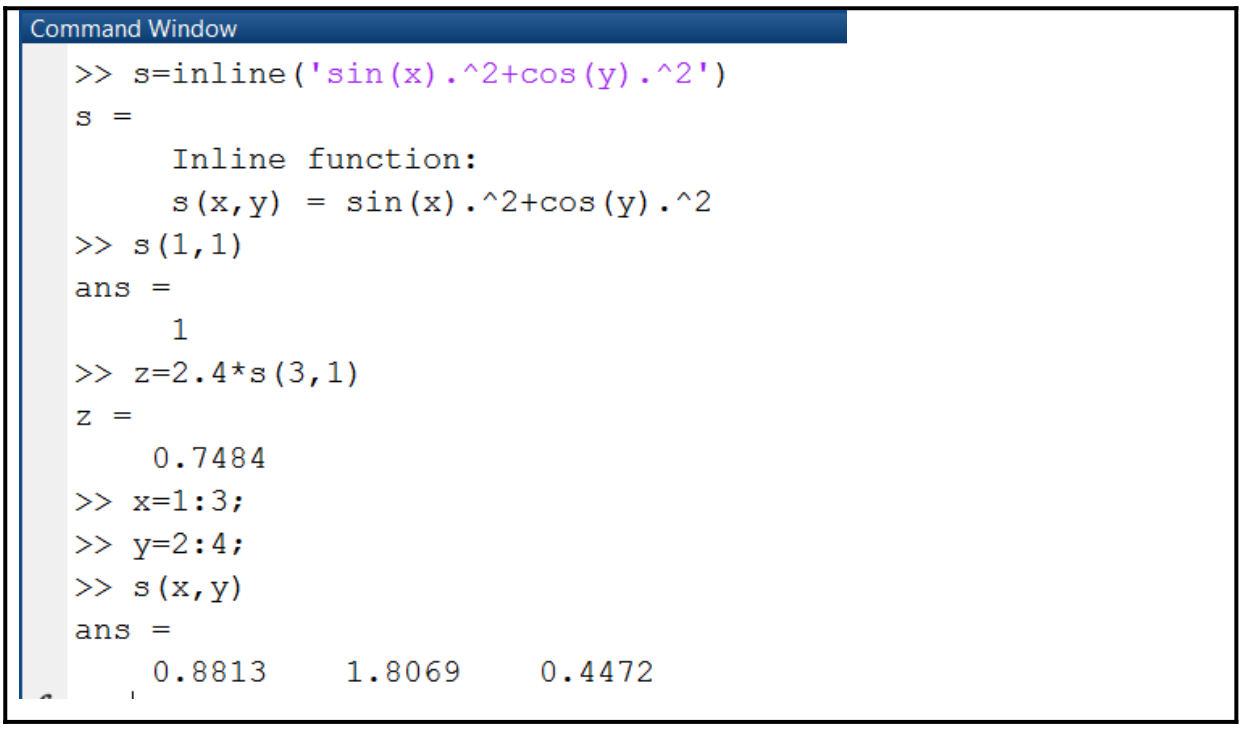


Рис. 1.1.2-4. Создание функции с помощью **inline()**

Наряду с рассмотренными выше функциями пользователя в Matlab имеется возможность создания функций (или некоторой последовательности вычислений) в виде **m-файлов**, которые можно сохранять и использовать в других сеансах работы. Подробнее материал о создании и работе с **m-файлами**, а также о средствах программирования в среде Matlab, изложен в **Теме 3.1**.

Символьная константа – это последовательность символов, заключенных в одиночные апострофы. Например, **'Кафедра ВМиП'**

Комментарии в Matlab определяются с помощью символа **%**. Например,

% Это комментарий

1.1.3. Лабораторная работа по теме «Элементы рабочей среды Matlab и простейшие вычисления»

1. Вопросы, подлежащие изучению

1) Элементы основного окна Matlab.
2) Окно панели **Command Window**.
3) Установка свойств среды системы Matlab.
4) Основные объекты системы Matlab.
5) Правила записи и вычисления арифметических выражений.
6) Работа с функциями пользователя, заданными в окне **Command Window**.
7) Окна **Workspace** и **Command History.**

2. Общее задание

1) ***Изучите материал Темы 1.1 (п.п. 1.1.1 – 1.1.2)***.
2) ***Настройте*** *стандартную конфигурацию* ***Рабочей среды****, выполнив команду* ***Default***.
3) ***Выполните команду clear all*** *для очистки* ***Рабочей среды***.
4) ***Выберите вариант*** *задания формул из* ***табл. 1.1.3-1***.
5) ***Задайте*** *переменным* ***x*** *и* ***y*** *допустимые числовые значения.*
6) ***Проанализируйте информацию****, возникшую в окне* ***Workspace.***
7) ***Введите формулу*** *для вычислений арифметического выражения и получите результат.*
8) ***Измените значения*** *исходных данных.*
9) ***Измените формат*** *вывода результата, выполнив команду* ***format*** *long, и* ***произведите перерасчет*** *значения выражения.*
10) ***Верните формат*** *вывода данных* ***short.***
11) ***Представьте*** *арифметическое выражение в виде правой части функции* ***f(x).***
12) ***Опишите функцию f(x)*** *с помощью оператора* ***@,*** *и* ***получите*** *вначале ее символьное выражение, а затем ее числовое значение при новом значении переменной* ***x***.
13) ***Опишите функцию f(x)*** *с помощью* ***inline().***
14) ***Измените значение*** *переменной* ***y***, ***вычислите*** *значения выражения* ***b*** *и функции* ***f(x)***.
15) ***Объясните****, почему изменение значения* ***y*** *привело к изменению значения* ***b,*** *но не повлияло на значение функции.*
16) ***Задайте диапазон изменения*** *аргумента функции с шагом, позволяющим получить таблицу значений функции* ***f(x)*** *для заданных значений аргумента (порядка 8-10 точек), и* ***выведете значения*** *функции* ***f(x)*** *в выбранном интервале вначале в строку, а затем в столбец.*

17) ***Выполните команду who*** *и* ***проанализируйте выведенную*** *информацию о данных.*
18) ***Выполните команду whos*** *и* ***проанализируйте*** *выведенную информацию о данных.*
19) ***Установите путь*** *к папке, находящейся на вашей флешь-карте, для сохранения содержимого окна* ***Command Window****.*
20) ***Создайте файл*** *в текущей папке.*
21) ***Выделите в окне Command Window****с использованием команд* ***diary on*** *и****diary off****область сохранения.*
22) ***Сохраните текст*** *рабочего окна на внешнем носителе.*
23) ***Представьте результаты*** *работы преподавателю, ответьте на поставленные вопросы.*
24) ***Выполните команду clear all*** *для очистки* ***Рабочей среды****.*
25) ***Оформите отчет*** *по выполненной работе.*

3. Варианты индивидуальных заданий

Таблица 1.1.3-1

№	Формулы для вычислений	№	Формулы для вычислений
1.	$t = \cos\frac{\pi}{7} * \frac{\sin^2(x - 8y)}{2{,}7(x - \pi)}$	**16.**	$b = \frac{\lg x - \sin^2 xy}{0{,}8 \cdot \ln(1 - x)^2}$
2.	$d = \frac{(1 - e^{xy})^2}{0{,}7\lg\lvert 1 - x^2\rvert}$	**17.**	$d = 10^4 \cdot \frac{e^{-\frac{x}{2y}} + \sqrt{\lvert \sin y^3 \rvert}}{2{,}5\cos^2 x}$
3.	$h = \frac{xy + \sin x}{\lvert 1 - y\rvert * \ln x}$	**18.**	$f = \frac{\frac{\pi}{3} + \ln x^3}{3y - x} + x \cdot \sin y^2$
4.	$c = \frac{(yx^2 - 1)^2}{2} \cdot (\cos^2 y - \sin x^2)$	**19.**	$h = \frac{208 \cdot \lg x + x^2}{\lvert x - y^2\rvert - e^{-y}}$
5.	$b = \sqrt[3]{\frac{x + y}{0{,}2x}} \cdot \sin(tg^2 x)$	**20.**	$a = 10^5 \cdot \lg\lvert 0.8x\rvert \cdot e^{\frac{-x^2}{2xy}}$
6.	$d = \frac{xe^{xy} + 8\sin^2 x}{x(x - y)(3x + y)}$	**21.**	$b = \frac{x^y}{1 - \frac{1}{e^{-x + \sin y}}}$
7.	$z = \frac{\pi}{2} - \sqrt{2x} - \frac{x + y^2}{0{,}75tg\lvert x + y\rvert}$	**22.**	$c = x \cdot \lg\lvert x - 6\rvert - \frac{\sin x^2}{yx^3}$
8.	$d = \frac{xy^2 - \sqrt{\lvert x^2 - 2{,}5 \cdot 10^{-3} y\rvert}}{2\sin xy} + 0.5$	**23.**	$a = \frac{14 \cdot \sin x + y^2}{0.92 \cdot \cos^3 x}$
9.	$f = 5{,}2^3 \cdot \frac{\lg(x + y)}{x - \frac{1}{0{,}45\sin(x - 8y)}} + 0.5$	**24.**	$a = \frac{x^2 - xy}{0.7\sin\ln\lvert x\rvert}$
10.	$a = 0{,}8 \cdot 10^{-5}(xe^{-x(y - 1{,}2)} - yx)^3$	**25.**	$c = \frac{2.71x^2 - \cos y}{tg(x^2) \cdot e^{-y}}$

11.	$d = \frac{\sqrt{\lvert x \rvert} + e^{-y}}{5{,}8 \cdot \cos y^3}$	**26.**	$d = \frac{1 - tgxy^2}{\sqrt[3]{x}} + 4\sqrt{x^2 - 0{,}1}$
12.	$f = - \frac{2x^2 - \sin x^2}{2 - e^{-y}}$	**27.**	$f = 0.5 + \frac{1}{2} \cdot \cos \frac{1 - \sin xy^2}{1 + \sin^2 xy}$
13.	$h = \frac{\sin^3 x + e^{-\sin y}}{0{,}6x^2y^2}$	**28.**	$g = x \cdot e^{-y} + \frac{(x+y)^2}{2 \cdot \cos^3 x}$
14.	$a = 10 \cdot \frac{\ln y^2 - \sqrt[4]{\lvert x - y \rvert}}{1 - \cos^3 y}$	**29.**	$z = \frac{x - y}{\sqrt{x+y}} + \frac{xy^2}{\sin x^2 \cdot \cos^2 y}$
15.	$c = \frac{1}{2\pi} - x\sqrt{2{,}5 \cdot 10^3 y} \cdot \lvert \cos x^3 \rvert$	**30.**	$b = \left\lvert \pi - \frac{x}{3} \right\rvert \cdot e^{\frac{1 - \sin e^{-y}}{2x}}$

4. Содержание отчета

1) В начале сессии введите в формате комментариев:
- название лабораторной работы;
- ФИО студента, номер группы;
- № варианта;
- индивидуальное задание.

2) Протокол вычислений (сессии) в окне **Command Window** в соответствии с общим заданием, снабженный подробными комментариями.

1.1.4. Контрольные вопросы по теме

1) Назначение окна **Command Window**.
2) Назначение окна **Command History**.
3) Окно отображения информации о данных.
4) Назначение команд **who** и **whos**.
5) Каким образом установить формат для вывода числовых значений на экран?
6) Каким образом перенести командную строку из окна **Command History** в окно **Command Windows**?
7) Каким образом установить формат для вывода числовых значений на экран?
8) Форматы числовых данных в системе Matlab.
9) Что такое системные константы?
10) Что такое символьные константы?
11) Команда объявления символьных переменных.
12) Какой символ используется для определения комментария?
13) Какая из операций .* или * является операцией поэлементного умножения?
14) Какой символ предназначен для запрета вывода результата выполнения команды на экран?
15) Создание функций с помощью команды @.
16) Создание функции с помощью **inline()**.

Тема 1.2. Векторы, матрицы и построение графиков в системе Matlab

1.2.1. Вектора и матрицы
1.2.2. Построение графиков и визуализация вычислений в системе MatLab
1.2.3. Лабораторная работа по теме
1.2.4. Контрольные вопросы по теме

1.2.1. Векторы и матрицы

Matlab построена как система, ориентированная на работу с матрицами, то есть все численные вычисления производятся в матричной форме. Даже обычные числа и переменные в Matlab рассматриваются как матрицы размера **1 x 1**. К особенностям работы с массивами в Matlab относится то, что одномерный массив может быть **вектором-строкой** или **вектором-столбцом (**рис. 1.2.1-1).

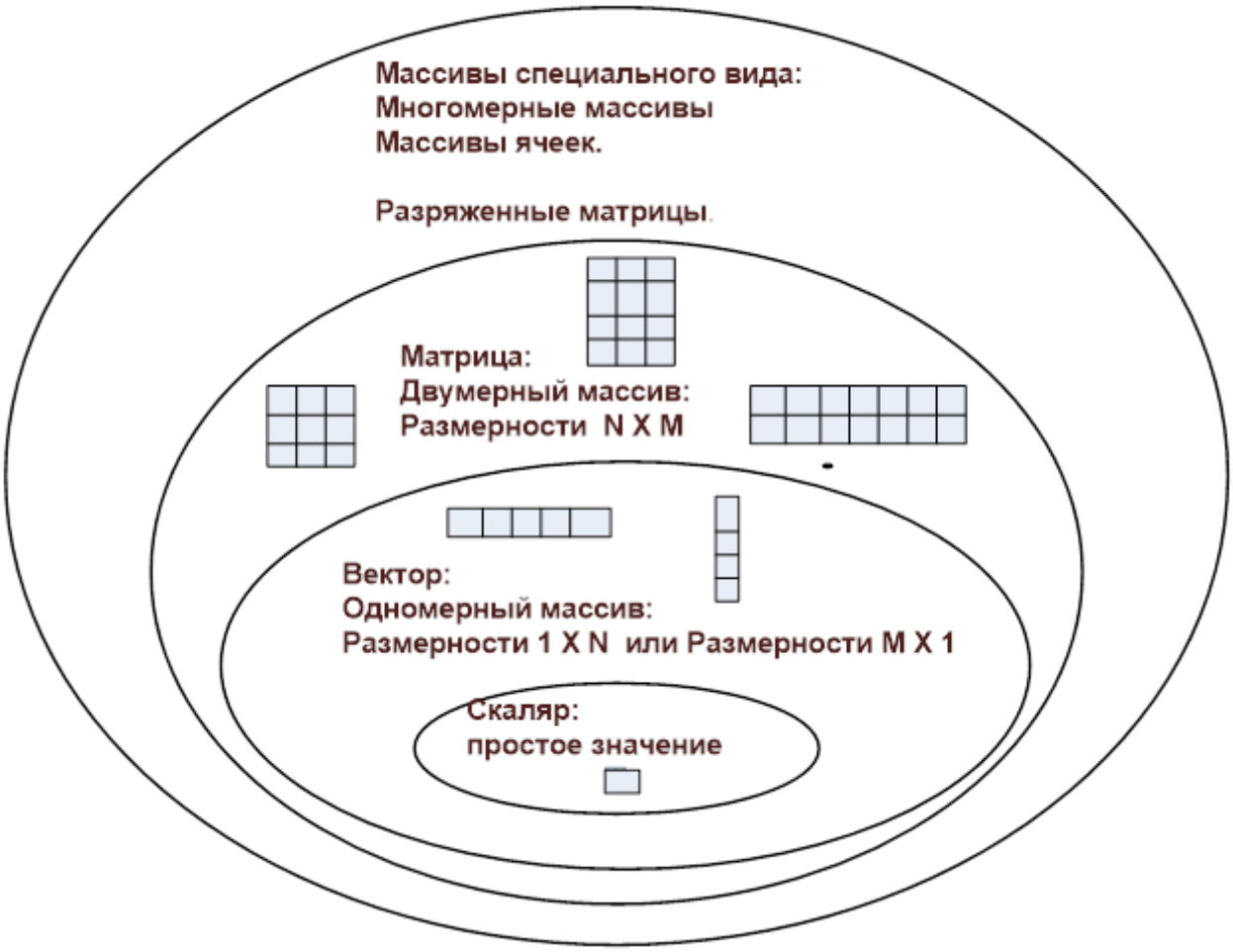


Рис. 1.2.1-1. Представление данных в Matlab: переменные, вектора (одномерные массивы) и матрицы (двумерные массивы)

Для определения вектора используется операция квадратные скобки (операция объединения), а элементы вектора (рис. 1.2.1-2) отделяются друг от друга:

- точкой с запятой, если нужно получить вектор-столбец;
- пробелом или запятой, если нужно разместить элементы в векторе-строке.

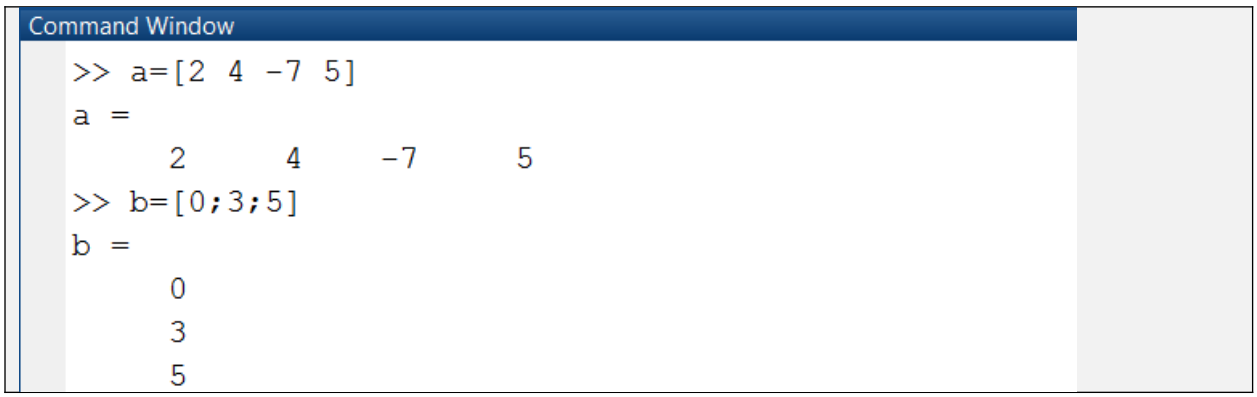


Рис. 1.2.1-2. Создание вектора-строки **a=[ ]** и вектора-столбца **b=[ ]**

Для определения длины вектора используется функция **length(a)**, где **a** – имя вектора, а для операции транспонирования используется символ апостроф (') (рис. 1.2.1-3).

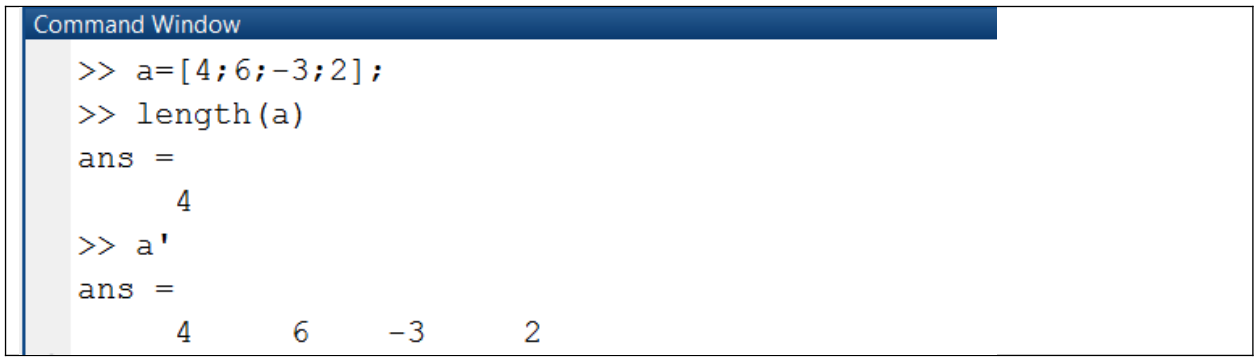


Рис. 1.2.1-3. Определение длины вектора и его транспонирование

В следующем примере (рис. 1.2.1-4) при описании вектора **x** символ двоеточие, поставленный между двумя числами, указывает, что его элементы последовательно принимают значения, начиная от первого числа (0) до последнего числа (5) с шагом **1** (по молчанию шаг равен 1). При описании вектора **y** использован шаг **0.1,** и выведены значения элементов вектора.

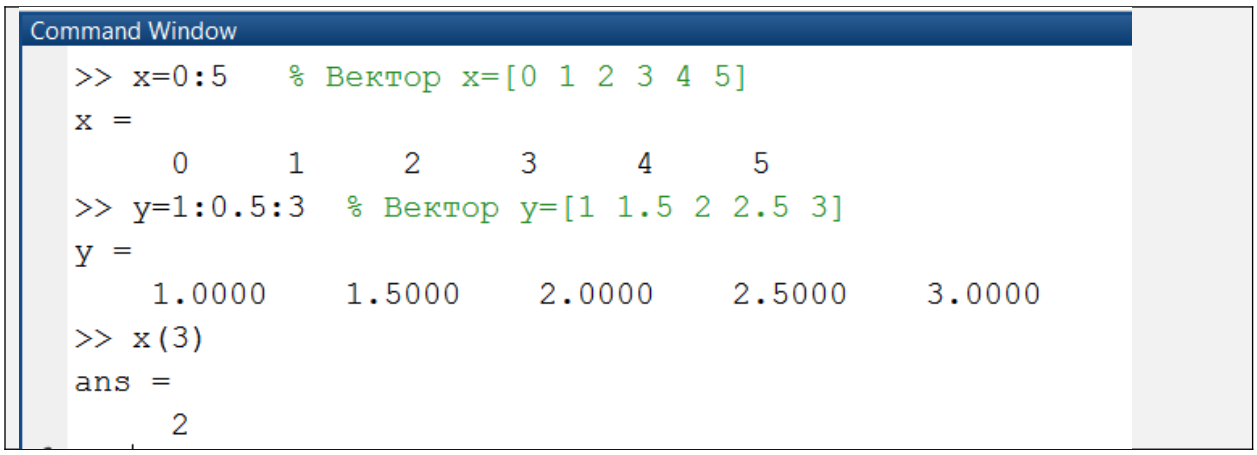


Рис. 1.2.1-4. Способы описания векторов с постоянным шагом

Для описания матрицы необходимо ввести ее имя и знак присваивания, а затем в квадратных скобках значения ее элементов. При этом значения элементов строк вводятся через пробел, а строки матрицы разделяет символ точка с запятой (**;**) (рис. 1.2.1-5):

A=[v1;v2;v3], где **v1, v2, v3** - векторы одинаковой размерности.

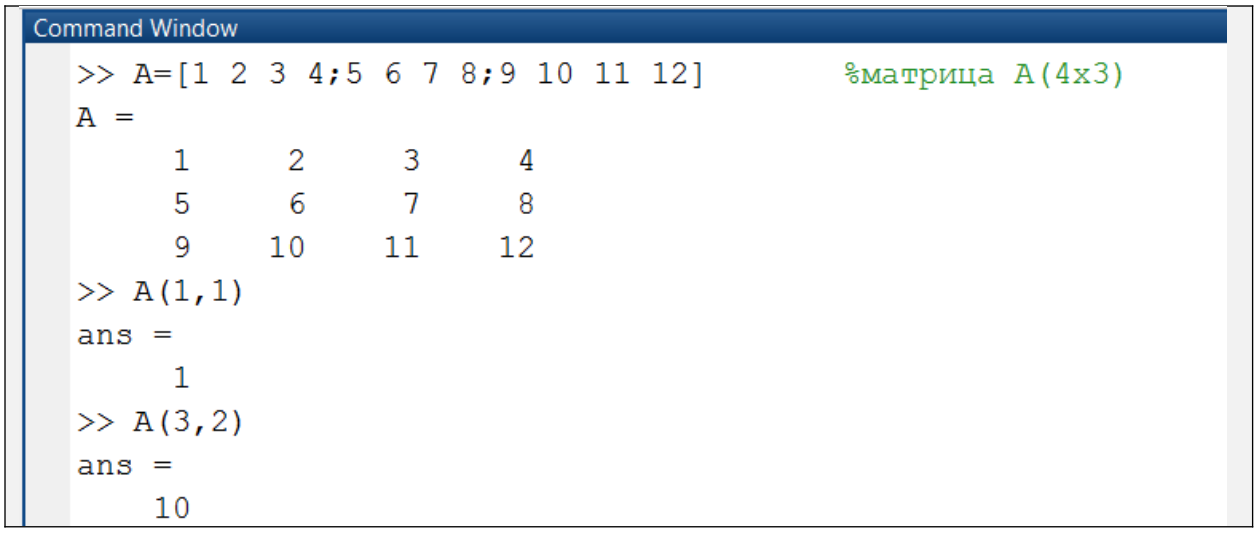


Рис. 1.2.1-5. Создание матрицы **A(3,4)** и доступ к ее элементам

Следует помнить, что нумерация элементов матрицы (в строках и столбцах) начинается с 1.

Matlab обладает большим набором встроенных функций для обработки векторных данных. Полный список имеющихся функций выводится в командное окно при помощи команды **help datafun**. Некоторые из них приведены в таблице 1.2.1-1.

Таблица 1.2.1-1

Функции операции над векторами	
prod(V)	Произведение элементов вектора
sum(V)	Сумма элементов вектора
min(V)	Минимальный элемент вектора
max(V)	Максимальный элемент вектора
mean(V)	Среднее значение элементов вектора
sort(V)	Сортировка элементов вектора по возрастанию (-sort(V) – по убыванию)
Функции определения матриц и операций над ними	
det(A)	Вычисляет определитель квадратной матрицы
rand ([n,m])	Возвращает матрицу, элементы которой распределены по равномерному закону
randn ([n,m])	Возвращает матрицу, элементы которой распределены по нормальному закону
size(A)	Определяет число строк и столбцов матрицы **A**, результат – вектор **[n;m]**
sum(A [,k])	Формирует вектор-строку (k – есть) или вектор-столбец (k- нет), каждый элемент которого – сумма элементов строки или столбца
min(A)	Формирует вектор-столбец из минимальных элементов строк
max(A)	Формирует вектор-столбец из максимальных элементов строк
sort(A)	Упорядочивает элементы столбцов по возрастанию
norm(A [,p])	Возвращает норму матрицы (по умолчанию вычисляется вторая норма)
inv(A)	Возвращает матрицу, обратную A

Примеры использования некоторых функций над векторами приведены на рис. 1.2.1-6.

```
Command Window
>> x=[6,1,-4,2];
>> prod(x)     %Произведение элементов вектора x
ans =
   -48
>> sum(x)     %Сумма элементов вектора x
ans =
     5
>> min(x)     %Минимальный элемент вектора x
ans =
    -4
>> max(x)     %Максимальный элемент вектора x
ans =
     6
>> mean(x)    %Среднее арифметическое значение элементов
ans =
    1.2500
>> sort(x)    %Сортировка по возрастанию
ans =
    -4     1     2     6
>> -sort(x)   %Сортировка по убыванию
ans =
     4    -1    -2    -6
```

Рис. 1.2.1-6. Примеры функции над векторами

```
Command Window
>> A=[-1 2 0;2 1 -1 ; 2 1 3]
A =
    -1     2     0
     2     1    -1
     2     1     3
>> det(A)         %Вычисления детерминанта (определителя) матрицы
ans =
   -20
>> rand(3,2)  %Матрица случайных чисел по равномерному закону распределения
ans =
    0.8147    0.9134
    0.9058    0.6324
    0.1270    0.0975
>> randn(3) %Квадрамная матрица с нормальным законом распределения
ans =
   -0.4336    2.7694    0.7254
    0.3426   -1.3499   -0.0631
    3.5784    3.0349    0.7147
```

Рис. 1.2.1-7. Примеры определения матриц

На рис. 1.2.1-7 приведены примеры вычисления определителя квадратной матрицы и заполнения матриц случайными числами, сгенерированных по равномерному и нормальному законам.

В следующем примере (рис. 1.2.1-8) показано использование функции Matlab **sum()**.

```
Command Window
>> A=[1 2 3;1 2 3]
A =
     1     2     3
     1     2     3
>> sum(A,1)  %Сумма элементов матрицы по столбцам
ans =
     2     4     6
>> sum(A,2)  %Сумма элементов матрицы по строкам
ans =
     6
     6
>> sum(sum(A))  %Сумма всх элементов матрицы
ans =
    12
```

Рис. 1.2.1-8. Варианты использования функции Matlab **sum()**

Пример, приведенный на рис. 1.2.1-9, демонстрирует функции, определяющие минимальные и максимальные значения матриц.

```
Command Window
>> A=[1 -2;8 4]
A =
     1    -2
     8     4
>> min(A)  %Наименьшие элементы столбцов
ans =
     1    -2
>> min(min(A))  %Наименьший элемент матрицы A
ans =
    -2
>> max(A)  %Наибольшие элементы столбцов
ans =
     8     4
>> max(max(A))   %Наибольший элемент матрицы A
ans =
     8
```

Рис. 1.2.1-9. Определение минимальных и максимальных значений

В примере, приведенном на рис. 1.2.1-10, показаны функции, позволяющие определять средние значения элементов в столбцах (или в строках) и проводить упорядочение (сортировку) элементов в строках (или столбцах).

```
Command Window
>> A=[3 5 1; 6 10 2;9 6 3]
A =
     3     5     1
     6    10     2
     9     6     3
>> mean(A)  %Средние значения по столбцам
ans =
     6     7     2
>> sort(A)   %Столбцы упорядочены по возрастанию
ans =
     3     5     1
     6     6     2
     9    10     3
```

Рис. 1.2.1-10. Определение средних значений и упорядочение элементов

Известно, что если детерминант матрицы отличен от нуля, то это невырожденная матрица. Для такой матрицы может быть вычислена обратная матрица (A^{-1}), которая при умножении на исходную матрицу A дает единичную (по диагонали расположены единицы, а прочие элементы равны нулю). Для получения обратной матрицы используется функция **inv().** Умножение матриц в Matlab производится только с использованием их имен. Описанные действия приведены на рис. 1.2.1-11.

```
Command Window
>> A=[1 2 3;2 -2 3;0 1 2]
A =
     1     2     3
     2    -2     3
     0     1     2
>> B=inv(A)
B =
    0.7778    0.1111   -1.3333
    0.4444   -0.2222   -0.3333
   -0.2222    0.1111    0.6667
>> A*B
ans =
    1.0000    0.0000         0
   -0.0000    1.0000         0
         0         0    1.0000
```

Рис. 1.2.1-11. Получение обратной и единичной матриц

Рассмотрим еще один пример, в котором матрица умножается на скаляр, матрица делится на скаляр и матрица умножается на вектор (рис. 1.2.1-12).

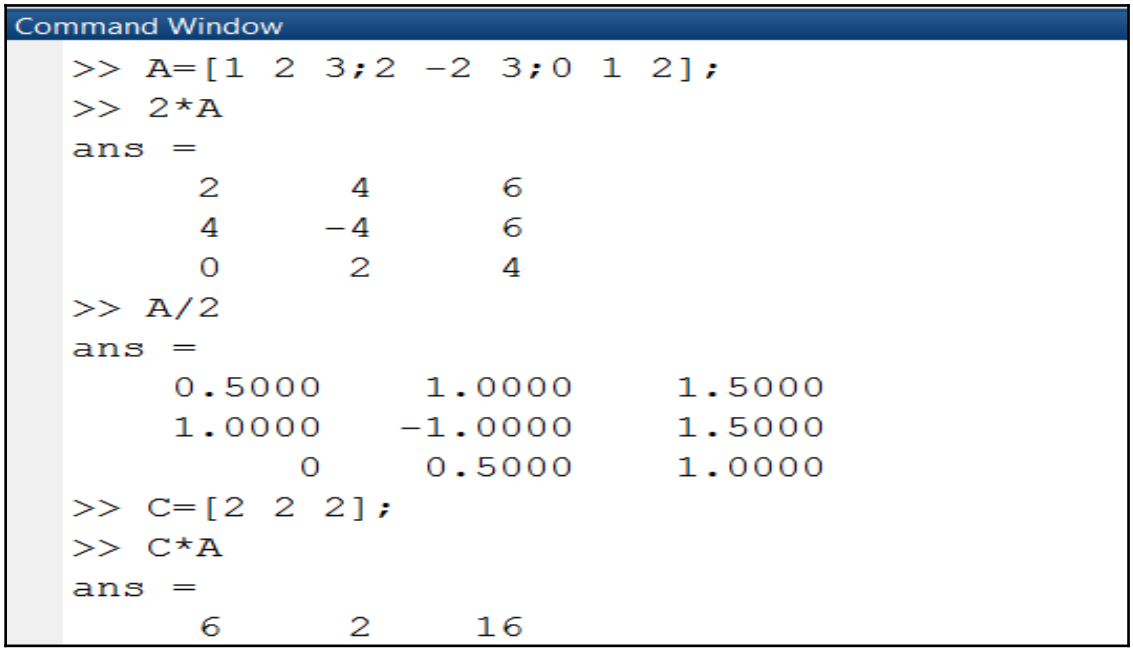


Рис. 1.2.1-12. Действия над матрицами

Вектора и матрицы кроме традиционного их применения для хранения и обработки данных необходимы и для построения графиков функций. При этом вектора используются для построения плоских графиков (графиков функций от одной переменной), а матрицы – для построения трехмерных изображений (графиков функций от двух переменных).

Обратите внимание, что после ввода или формирования элементов векторов и матриц могут возникнуть ошибки. Для контроля и исправления отдельных элементов векторов и матриц можно воспользоваться окном редактора данных (рис. 1.2.1-13).

Окно редактора массива данных (**Variables** – *ИмяПеременной*) состоит из панели инструментов и области просмотра значений переменных. В окне редактора данных (рис. 1.2.1-13) можно отображать несколько переменных. Переключение между переменными реализуется с помощью вкладок.

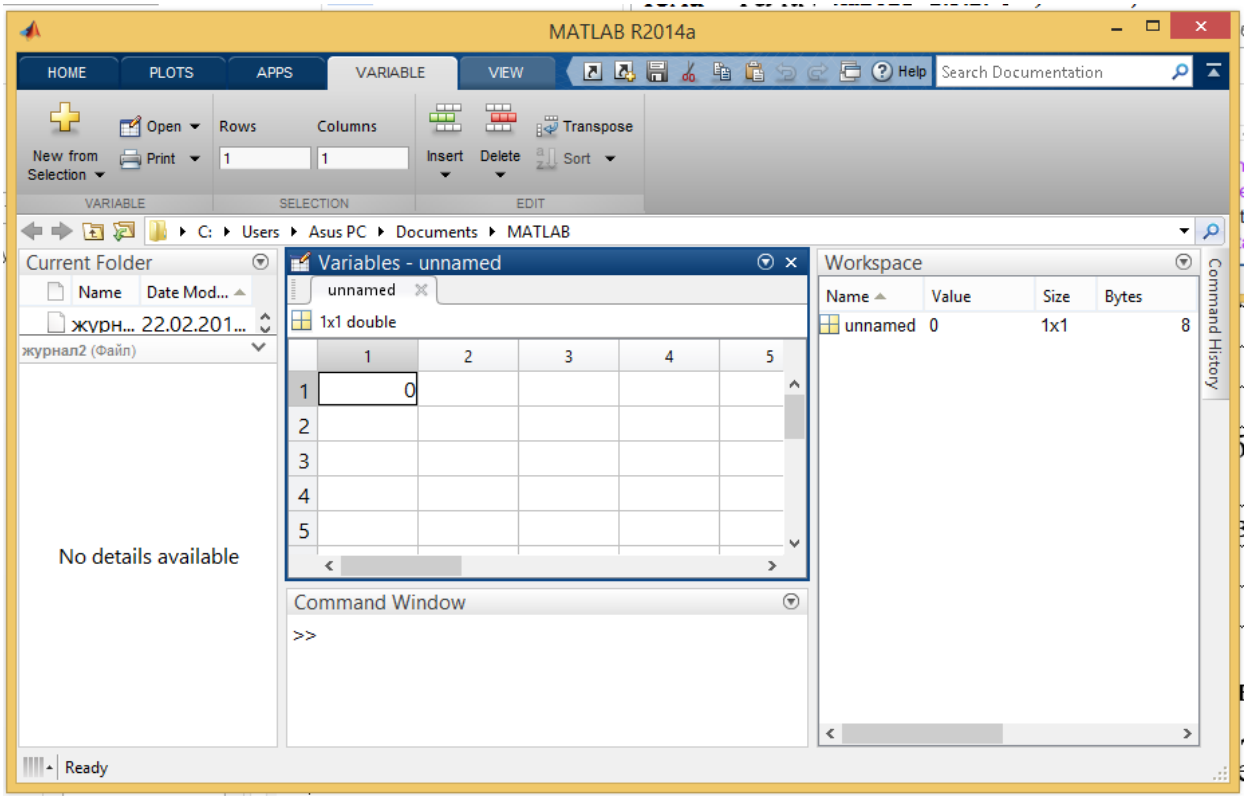


Рис. 1.2.1-13. Окно **Рабочей среды** с активной инструментальной панелью **VARIABLE** и открытым **Редактором данных**

1.2.2. Построение графиков и визуализация вычислений в системе MatLab

Самый простой способ для построения графика функции одной переменной **y=f(x)** - это предварительное формирование двух векторов одинаковой длины: вектора значений аргументов **x** и вектора соответствующих им значений функции **y**, а затем выполнение команды **plot(x, y)**. Выполнение команды **plot(x, y)** открывает графическое окно и отображает в нем график функции **y(x)**.

Рассмотрим пример построения графика функции $y = e^{-x^2}$ на отрезке [-1.5;1.5] (рис. 1.2.2-1).

```
Command Window
>> x = - 1.5:.01:1.5;
>> y=exp(-x.^2);
>> plot(x,y)
```

Рис. 1.2.2-1. Команды построения графика функции $y = e^{-x^2}$

В результате выполнения команды **plot(x,y)** появляется графическое окно с именем **Figure 1** (рис. 1.2.2-2).

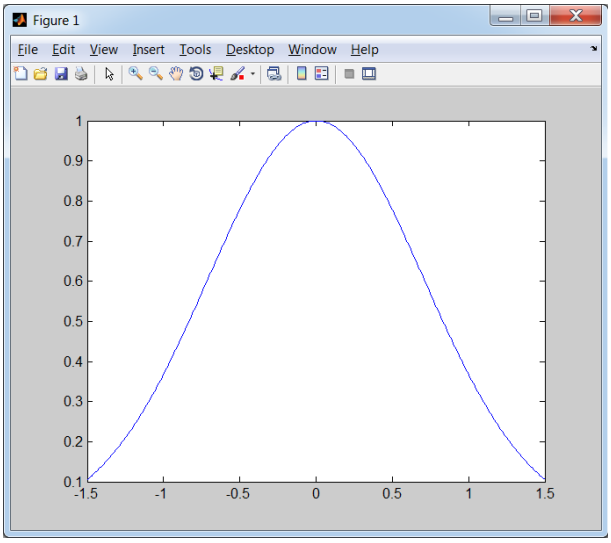


Рис. 1.2.2-2. Графическое окно с изображением графика функции $y = e^{-x^2}$

Переход между окнами (от командного окна к графическому окну и обратно) осуществляется с помощью комбинации клавиш <Alt+Tab> или с помощью мыши.

В общем случае, число аргументов у функции **plot()** не ограничивается двумя. Эта функция имеет следующий формат:

plot(x1,y1,x2,y2,...).

Таким образом, в одном графическом окне можно построить не один, а несколько графиков. Рассмотрим два способа построения в одном графическом окне трех графиков (рис. 1.2.2-3 и рис. 1.2.2-4).

```
Command Window
>> x=0:.01:2*pi;
>> y1=sin(x);
>> y2=sin(2*x);
>> y3=sin(4*x);
>> plot(x,y1,x,y2,x,y3)
```

Рис. 1.2.2-3. Использование векторов значений функций для построения графиков трех функций

```
Command Window
>> x=0:.01:2*pi;
>> y=[sin(x)',sin(2*x)',sin(4*x)'];
>> plot(x,y)
```

Рис. 1.2.2-4. Использование матрицы значений функций для построения графиков трех функций

Приведенные в первом и втором примерах наборы команд позволяют получить один и тот же результат (рис. 1.2.2-5.) Разница в том, что в первом

примере формируются три вектора значений функций (**y1**, **y2**, **y3**), а во втором – матрица **y**, содержащая значения функций в виде столбцов.

График, выведенный в графическое окно Matlab, может быть снабжен ***заголовком***, ***именами осей***, дополнительным ***текстом, сеткой*** и другой поясняющей информацией. Аргументами команд, управляющими пояснениями, являются текстовые строки. Например, команда **title()** добавит к графику заголовок. Команда **grid on** позволяет отобразить координатную сетку. Для вывода подписи осей используются команды **Xlabel()**, **Ylabel()**.

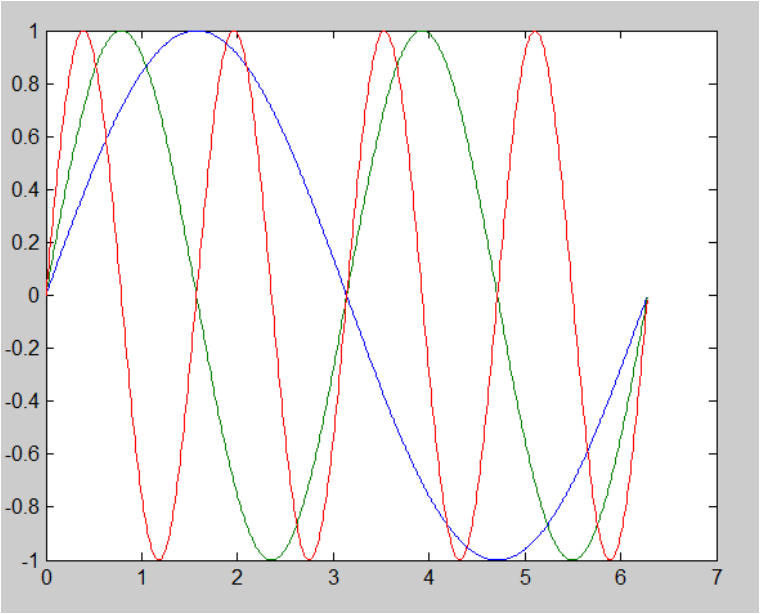


Рис. 1.2.2-5. Графики функций **y 1=sin(x); y2=sin(2*x); y3=sin(4*x)**

Если в одном графическом окне создаются несколько графиков, полезно с использованием функции **legend()** отобразить легенду графиков. Ее текстовые аргументы соответствуют подписям соответствующих созданию кривых, а числовой параметр определяет место расположения в графическом окне (табл. 1.2.2-1).

Таблица 1.2.2-1

Значение аргумента	**Размещение легенды**
-1	В правом верхнем углу над областью графика
0	Место выбирается автоматически, чтобы не перекрывать область кривых
1	В правом верхнем углу (и по умолчанию)
2	В левом верхнем углу области графика
3	В левом нижнем углу области графика
4	В правом нижнем углу области графика

При выводе в одно графическое окно нескольких графиков удобнее каждый график выводить с использованием своей команды **plot(),** однако в

этом случае происходит создание нового графического окна. Для того чтобы этого избежать, используется команда **holdon**. Эта команда позволяет расположить все в дальнейшем выводимые графики в одном окне.

При выводе графика можно сменить принятый по умолчанию Matlab цвет и тип точек, с помощью которых рисуется данный график (табл.1.2.2-2). Символы, указывающие на цвет и тип точки, заключаются в апострофы и указываются в команде **plot()** после имени функции.

Таблица 1.2.2-2

Символ	**Цвет**	**Символ**	**Маркер**
y	желтый	**.**	точка
m	фиолетовый	**кружок**	:
c	голубой	**x**	x-метка
r	красный	+	плюс
g	зеленый	*	звездочка
b	синий	**s**	квадрат
w	белый	**d**	алмаз
k	черный	**v**	треугольник(вниз)
€	треугольник (вверх)	**-**	сплошная
<	треугольник (влево)		точечная
>	треугольник (вправо)	**-.**	штрих - пунктирная
p	шестиугольник	**-**	пунктирная
h	восьмиугольник		

Рассмотрим пример (рис. 1.2.2-6), в котором используются перечисленные выше опции, инструкции и функции. Результат выполнения инструкций, комментирующих графики, приведен на рис. 1.2.2-7.

Command Window

```
>> x=0:0.01:2*pi;
>> y1=sin(x);
>> y2=sin(2*x);
>> y3=sin(4*x);
>> plot(x,y1,'-k')
>> grid on
>> hold on
>> plot(x,y2,'b--')
>> plot(x,y3,'r.-')
>> title('Построение графиков трех функций')
>> xlabel('x')
>> ylabel('y1(x),y2(x),y3(x)')
>> legend('y1(x)','y2(x)','y3(x)',0)
```

Рис. 1.2.2-6. Использование инструкций при построении графиков

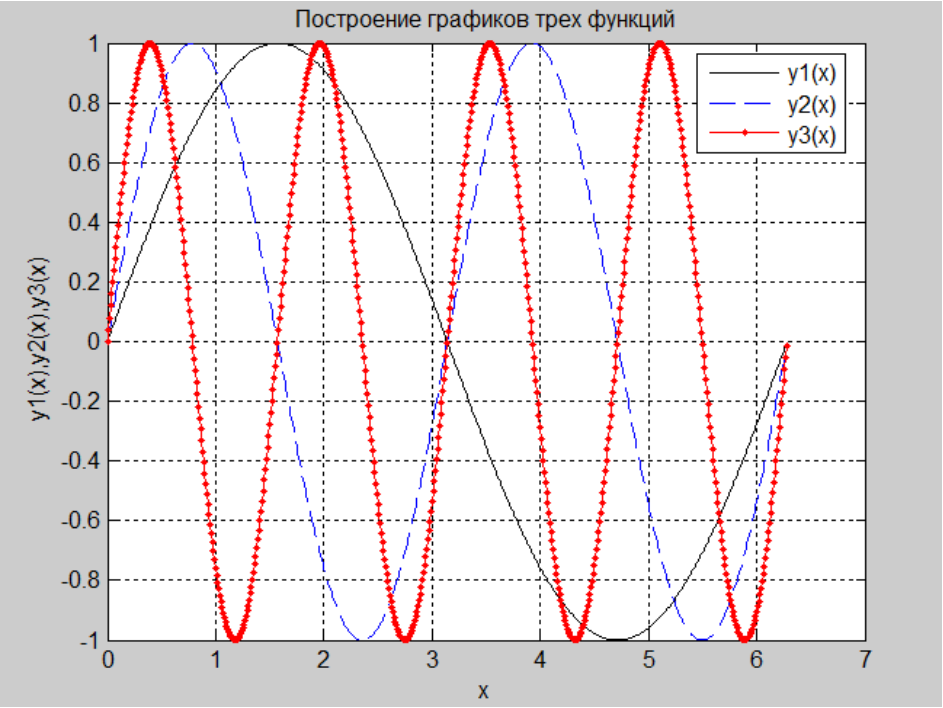


Рис. 1.2.2-7. Результат выполнения инструкций, комментирующих графики

Большую часть вышеописанных действий можно реализовать также с помощью команд инструментальной панели графических окон (рис. 1.2.1-8).

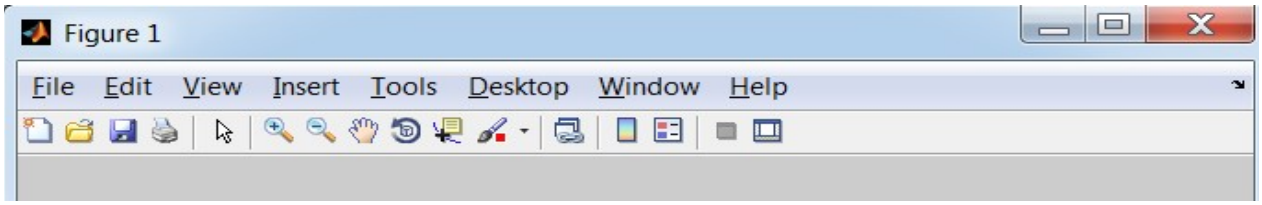


Рис. 1.2.2-8. Компоненты меню графического окна

Команды основного меню графического окна представляют большой набор средств, предназначенных для отображения и оформления графиков, позволяющих в интерактивном режиме и без использования команд придать графику желаемый вид, а кнопки панели дублируют наиболее часто используемые пункты меню, ускоряя тем самым доступ к ним.

Трехмерные поверхности обычно описываются функцией двух переменных **z(x,y).** Для построения трехмерных графиков необходимо сформировать два двумерных массива, например, **X** и **Y** с использованием функции **meshgrid() (**рис. 1.2.2-9).

```
Command Window
>> [X,Y]=meshgrid(0:3,-3:0);
>> X
X =
     0     1     2     3
     0     1     2     3
     0     1     2     3
     0     1     2     3
>> Y
Y =
    -3    -3    -3    -3
    -2    -2    -2    -2
    -1    -1    -1    -1
     0     0     0     0
```

Рис. 1.2.2-9. Формирование двумерных массивов функцией **meshgrid()**

```
Command Window
>> [x,y]=meshgrid(-4:4,-4:4);
>> f=@(x,y)x.^2+2*y.^2
f =
    @(x,y)x.^2+2*y.^2
>> z=f(x,y)
z =
    48    41    36    33    32    33    36    41    48
    34    27    22    19    18    19    22    27    34
    24    17    12     9     8     9    12    17    24
    18    11     6     3     2     3     6    11    18
    16     9     4     1     0     1     4     9    16
    18    11     6     3     2     3     6    11    18
    24    17    12     9     8     9    12    17    24
    34    27    22    19    18    19    22    27    34
    48    41    36    33    32    33    36    41    48
>> mesh(x,y,z)
>> plot3(x,y,z)
>> surf(x,y,z)
>> [Cm,h]=contour(x2,y2,z);
>> clabel(Cm, h)
>> grid on
>> surfc(x,y,z)
```

Рис. 1.2.2-10. Построение различных видов графиков функций двух переменных

Сформированные в виде двумерных массивов данные используются функциями:

- **mesh(X,Y,Z)** – построение сетчатого графика;
- **contour(X,Y,Z)** – построение графика контурных линий;
- **surf(X,Y,Z)** – построения графика сплошной поверхности;
- **surfc(X,Y,Z)** – построения графика сплошной поверхности и контурных линий;
- **plot3(X,Y,Z) –** построение точек, соединенных отрезками прямых и др.

Рассмотрим примеры использования перечисленных выше функций, для чего сформируем матрицу **z(x,y)** с использованием функции **f(x,y)** (рис. 1.2.2-10).

Результатом выполнения команды **mesh(x,y,z)** является построение графика поверхности в виде сетки (рис. 1.2.2-11).

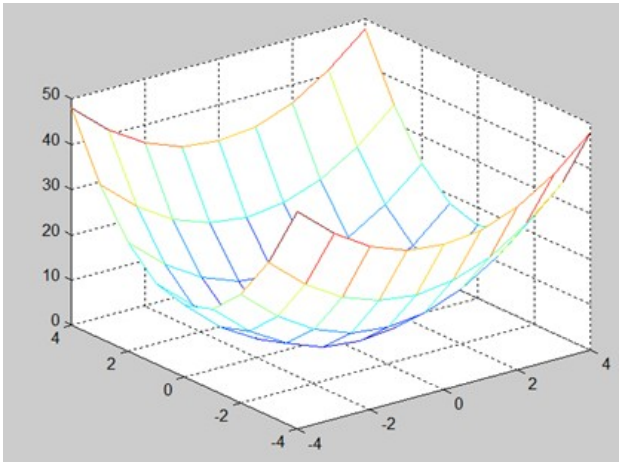


Рис. 1.2.2-11. Результат выполнения команды **mesh(x,y,z)**

В результате выполнения команды **plot3(x,y,z)** происходит построение графика поверхности, где точка соединены отрезками прямой (рис. 1.2.2-12).

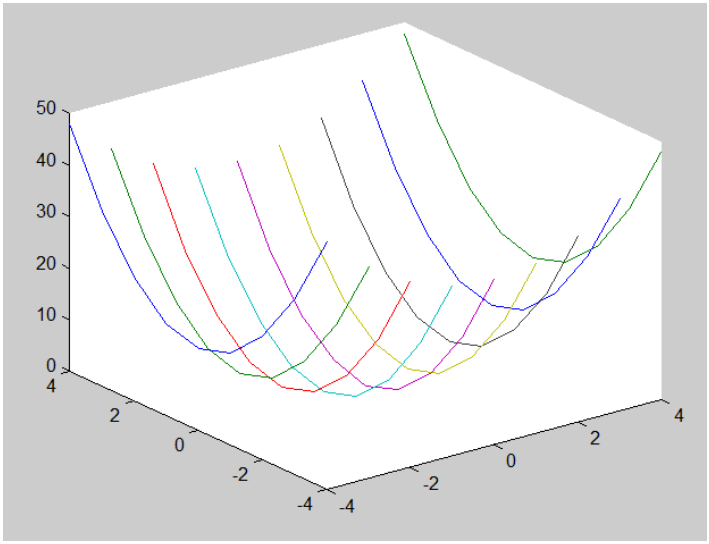


Рис. 1.2.2-12. Результат выполнения команды **plot3(x,y,z)**

Команда **surf(x,y,z)** выполняет построение графика сплошной поверхности (рис. 1.2.2-13).

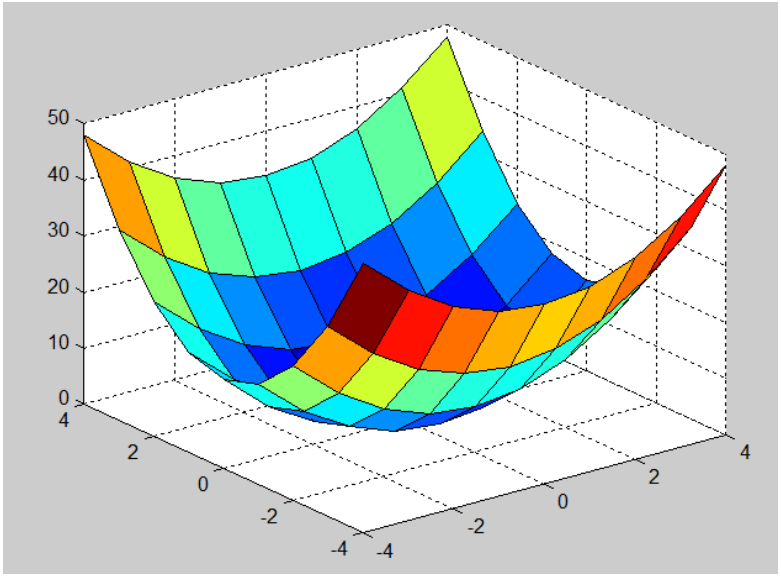


Рис. 1.2.2-13. Результат выполнения команды **surf(x,y,z)**

Команда **contour(x,y,z)** позволяет получить график контурных линий (рис. 1.2.2-14).

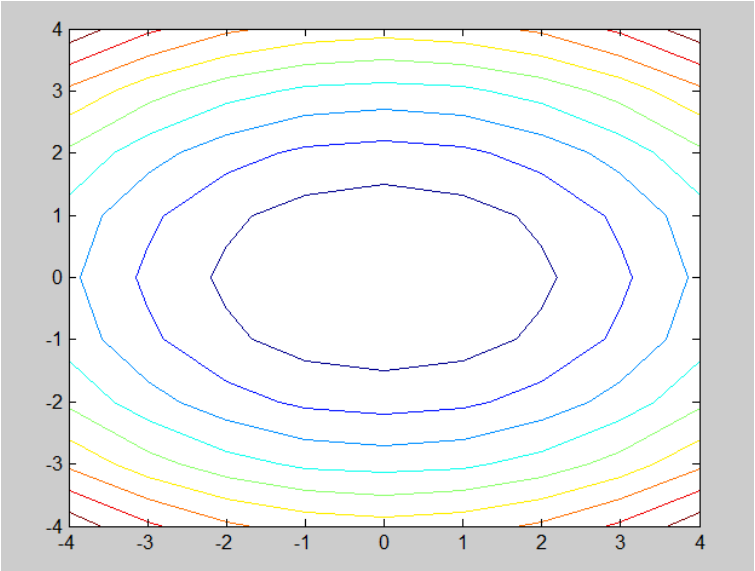


Рис. 1.2.2-14. Результат выполнения команды **contour(x,y,z)**

```
Command Window
>> [Cm,h]=contour(x2,y2,z);
>>clabel(Cm, h)
>>grid on
```

Рис. 1.2.2-15

Добавление команд, представленных на рис. 1.2.2-15, позволяет нанести на контурные линии значения функции в отдельных точках (рис. 1.2.2-16).

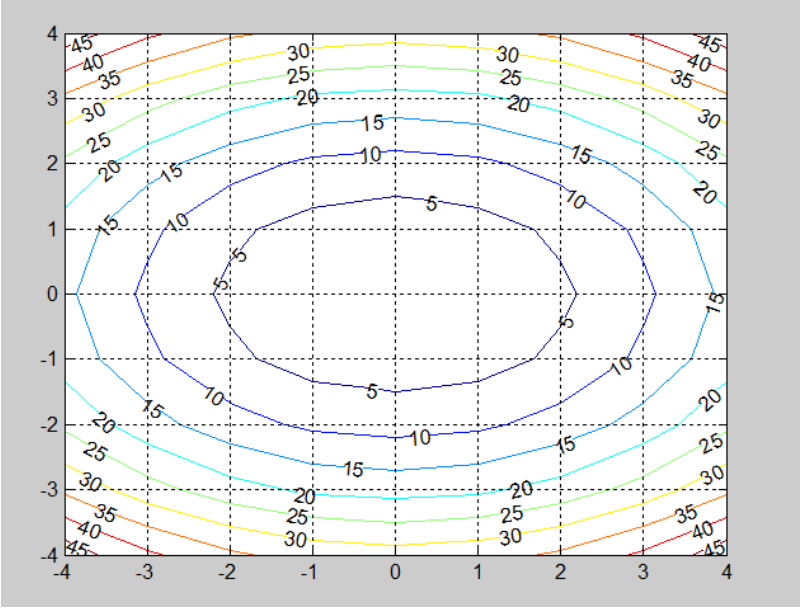


Рис. 1.2.2-16. График контурных линий с нанесенными значениями функции

Команда **surfc(x,y,z)** предназначена для построения графика сплошной поверхности, дополненного контурными линиями (рис. 1.2.2-17).

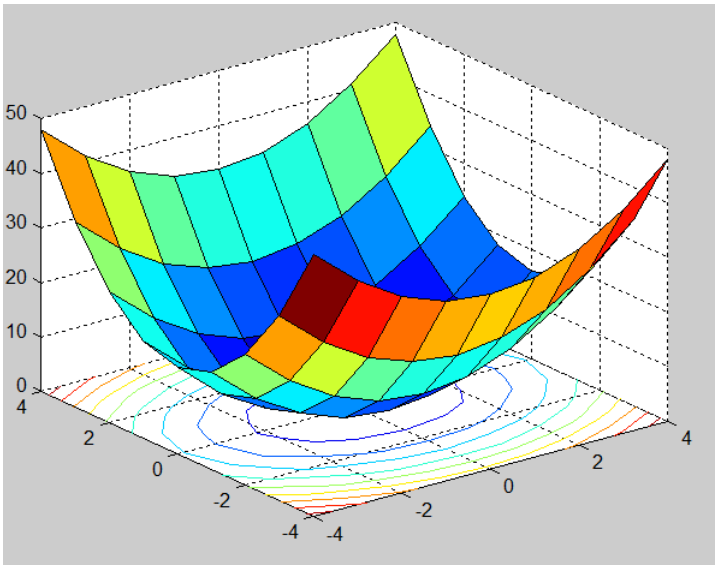


Рис. 1.2.2-17. Результат выполнения команды **surfc(x,y,z)**

1.2.3. Лабораторная работа по теме «Векторы, матрицы и построение графиков в системе Matlab»

1. Вопросы, подлежащие изучению

1) Работа с векторами и матрицами.
2) Построение графиков функций одной переменной**.**
3) Средства инструментальной панели графических окон.
4) Построение трехмерных изображений с использованием функций Matlab **mesh( ), plot3( ), surf( ), surfc( )** и **controur( )**.

2. Общее задание

1) ***Изучите*** *материал* ***Темы 1.2 (п.п. 1.2.1 – 1.2.2)****.*
2) ***Выберите вариант*** *индивидуального задания из* ***табл. 1.3.3-1****.*
3) ***Выполните команду clear all*** *для очистки Рабочей области.*
4) ***Опишите функцию f1(x)*** *и получите ее символьное выражение.*
5) ***Задайте диапазон*** *изменения аргумента функции* ***f1(x)*** *и вычислите ее значения.*
6) ***Задайте диапазон*** *изменения аргумента функции* ***f1(x)*** *для построения графика.*
7) ***Выполните команду plot( )****для получения графика* ***f1(x).***
8) ***Опишите функцию f2(x).***
9) ***Разместите графики*** *функций* ***f1(x)*** *и* ***f2(x)*** *в одном окне, для чего после построения графика первой функции выполните команду* ***hold on.***
10) ***Дополните графики*** *необходимыми пояснениями: заголовок, имена осей, координатная сетка и легенда.*
11) ***Задайте диапазоны*** *изменения значений* ***x*** *и* ***y*** *для функций* ***f3(x,y)*** *и получите таблицы их значений.*
12) ***Опишите функцию f3(x, y).***
13) ***Получите таблицу*** *значений функции* ***f3(x,y).***
14) ***Получите график функции f3(x, y)*** *с использованием команд* ***mesh( ), plot3( ), surf( ), surfc( )*** *и* ***controur( ).***
15) ***Сохраните текст*** *рабочего окна на внешнем носителе.*
16) ***Представьте результаты*** *работы преподавателю, ответьте на поставленные вопросы.*
17) ***Выполните команду clear all*** *для очистки Рабочей среды.*
18) ***Оформите отчет*** *по выполненной работе****.***

1. Варианты индивидуальных заданий

Таблица 1.3.3-1

1	$f1(x)=Sin(1-0.2x^2)-x$ $f2(x)=e^{-x}-2$ $f3(x,y)=x^2+y^2-8\sin(x-y)\cdot\cos(x+y)$
2	$f(x)=x-Sin(1/x)$ $f2(x)=e^x+\ln(x)-x$ $f3(x,y)=x^2-2y^2-\sin^2(x+y)-0.5e^{x\cdot\sin(y)}$
3	$f1(x)=1-x+Sinx-\ln(1+x)$ $f2(x)=(1-x)^{1/2}-Cos(1-x)$ $f3(x,y)=x^3-2y^2-\sin(x+y)\cdot\cos(x-y)-x+2y$
4	$f(x)=Sinx^2+Cosx^2-10x$ $f2(x)=x^2-\ln(1+x)-3$ $f3(x,y)=x^3-2y^2-\sin(x+y)\cdot\cos(x-y)-x+2y$
5	$f(x)=Cos(x/2)\cdot\ln(x-1)$ $f2(x)=Cos(x/5)\cdot(1+x)^{1/2}-x$ $f3(x,y)=x^3-2y^3-8\sin(x+y)\cdot\cos(x-y)-5x$
6	$f1(x)=3x-e^{-x}$ $f2(x)=4(1+x^{1/2})\ln(x)-10$ $f3(x,y)=3x^3-y^2-\cos(x-y)-x+2x-9xy$
7	$f1(x)=Sinx-3^{1/2}Cosx+4x-4$ $f2(x)=x-1/(3+Sin(3.6x))$
8	$f1(x)=0,25x^3+Cos(x/4)$ $f2(x)=2-x-\ln(x)$ $f3(x,y)=3x^3-y^2-\cos(x-y)-x+2y-9xy-e^{x\cdot\cos(y)}$
9	$f1(x)=x^2+4Sin(x)$ $f2(x)=tg(0,36x+0,4)-x^2$ $f3(x,y)=3x^3-y^2-\cos(x-y)-x+2y-9xy-e^{x\cdot\cos(y)}$
10	$f1(x)=1+\lg(x)-0,5$ $f2(x)=2\lg(x)-x/2$ $f3(x,y)=3x^3+y^2-\cos(x-y)-x+2y-9xy$
11	$f1(x)=X-\sin(x)-0,25$ $f2(x)=\lg(0,4x+0,4)-x^2$ $f3(x,y)=x^3+y^2-x\cdot\sin^2(x+y)+2y-9xy$

12	$f1(x)=\sqrt{x}-Cos(0,387x)$ $f2(x)=\lg(x)-7/(2x+6)$ $f3(x,y)=x^3+y^3\sin^2(x+y)+2y-9xy$
13	$f1(x)=tg(0,5x+0,2)-x^2$ $f2(x)=3x-Cos(x)-1$ $f3(x,y)=x+y^3\sin^2(x+y)+2y-9x$
14	$f1(x)=x-\lg x-0,5$ $f2(x)=1,8x^2-Sin(10x)$ $f3(x,y)=x+y^3\sin^2(x+y)+2y-9x^2$
15	$f1(x)=ctg(1,05x)-x^2$ $f2(x)=x\cdot\lg x-1.2$ $f3(x,y)=5x^2+y^3+2y-9x-0.5\cdot e^y$
16	$f1(x)=\sqrt{\lg(x+2)}$ $f2(x)=Sin0.5x+2-(x/2)^2$ $f3(x,y)=5x^2-y^2+2y-9xe^{sin(x)}$
17	$f1(x)=0,5x+\lg(x-1)-2$ $f2(x)=Sin(0,5+x)-2x+0,5$ $f3(x,y)=5x^2-7y^2\cdot\cos(x+y)+2y-9xe^{sin(x)}$
18	$f1(x)=\ln(x/6)+\sqrt{x}$ $f2(x)=\log_2(x)-1/(x+2)$ $f3(x,y)=-15x^2-7y^2\cdot\cos(x+y)+2x-9x+6$
19	$f1(x)=\lg(2+x)+x^2-3$ $f2(x)=\ln(1+2x)-2+x$ $f3(x,y)=15x^2+7y^2\cdot\cos(x+y)+2y-9x\sqrt{\lvert y\rvert}+6$
20	$f1(x)=e^{-x}-2+x^2$ $f2(x)=2e^x-x^2+2$ $f3(x,y)=-3x^2+y\cdot\cos(x+y)+2y^4-9x\cdot\sqrt{\lvert 3x^2-y\rvert}$
21	$f1(x)=2\cdot x^2-e^{x/2}$ $f2(x)=2\cdot arqtg(x)-3x+2$ $f3(x,y)=-5x^2+y^3+2x^3y-x^3\cdot\sqrt[3]{\lvert 3x-y\rvert}$
22	$f1(x)=Sin(x-0,5)-x-0,8$ $f2(x)=(x-3)^2\lg(x-2)+2$ $f3(x,y)=-5x^2-y^3+2x^3y-x^3\cdot\sin(3x-y)$
23	$f1(x)=x-3+Cosx+x^2$ $f2(x)=(x-1)\lg(x+11)-1$ $f3(x,y)=-5x^2-y^3+2x^3y-x^3\cdot\sin(3x+y)$

24	$f1(x)=e^{2x}Cos(2x)+x$ $f2(x)=x^2Cos(2x)+1$ $f3(x,y)=-5x^2+y^3+2x\cdot y-3\cdot\sin(3x-y^3)+8y\cdot\cos(x)$
25	$f1(x)=(2-x)2^x-1$ $f2(x)=(x-2)^2-1-2^x$ $f3(x,y)=x^2+y^3+2x\cdot y-3\cdot\sin(3x-y^3)+5y\cdot\cos(x)\cdot e^x$
26	$f1(x)=e^x+x+1$ $f2(x)=0,5^x-3-(x+2)^2$ $f3(x,y)=-5x^2+8y^2+2x\cdot y-10\cdot\sin(3x-y^3)+5y\cdot x$
27	$f1(x)=(x-2)^2\lg(x+5)-1$ $f2(x)=(x-1)^2\lg(x-3)-1$ $f3(x,y)=-5x^2+8y^2+2x^2\cdot y^2-10\cdot\sin(x-y^3)+5x^3\cdot\cos(x+y)^3$
28	$f1(x)=2x^2-2^x-20$ $f2(x)=x\log_3(x+1)-1$ $f3(x,y)=-5x^3-y^3+2x^2\cdot y-7(x\cdot y)^2\sin(3x)$
29	$f1(x)=0,5^x-3+(x+1)^2$ $f2(x)=2\,arctg(x)-x+3$ $f3(x,y)=e^x(5x^2-8y^2)+2x^2\cdot y^2-10\cdot\sin(x)\cdot\cos(10y)$
30	$f1(x)=5^x-6x-3$ $f2(x)=2Cos(x)+x^2-3x+2$ $f3(x,y)=5x^2-y^3+2x^3\cdot y^2-x\cdot\sin(3x-y)+y^2\cdot\cos(x)$

3. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в окне **Command Window**, снабженный соответствующими комментариями.

1.2.4. Контрольные вопросы по теме

1) Как создать вектор-строку?
2) Как создать вектор-столбец?
3) Как транспонировать векторы?
4) Какая функция служит для определения длины вектора?
5) Каким образом создать вектор с постоянным шагом?
6) Требуется ли при работе с векторами и матрицами предварительное объявление их размера?
7) Какой символ используется для разделения элементов матрицы в строке, а какой для разделения ее строк?
8) Какие команды предназначены для заполнения матрицы случайными числами, распределенными по равномерному или нормальному закону распределения?
9) Формат команд выбора минимального и максимального значения элемента матрицы.
10) Назначение команды **plot().**
11) Каким образом построить в одном графическом окне несколько графиков?
12) Какой пояснительной информацией может быть снабжен график, построенный в графическом окне?
13) Для чего используется функция **legend()?**
14) Каково назначение функции **meshgrid()** при построении трехмерных изображений?
15) Какие типы графиков позволяют строить встроенные функции: **contour(), surf(), survc()** и **plot3()**?

Тема 1.3. Средства Matlab для создания и описания m-файлов

1.3.1. Основные понятия и средства программирования в MatLab
1.3.2. Описание и работа с m-сценариями
1.3.3. Описание и работа с m-функциями
1.3.4. Основные операторы m-языка и программирование в MatLab
1.3.5. Примеры решения задач средствами MatLab
1.3.6. Лабораторная работа по теме
1.3.7. Контрольные вопросы по теме

1.3.1. Основные понятия и средства программирования в MatLab

Использование системы Matlab только в режиме непосредственного расчета (в командном режиме) явно недостаточно для решения серьезных задач, поскольку, во-первых, зачастую требует выполнения сложных алгоритмических процессов, а, во-вторых, необходим механизм хранения в библиотеках команд и операторов системы Matlab. То есть необходимы средства, какие есть в языках программирования высокого уровня.

Такие средства в Matlab существуют. Они состоят из так называемых **m**-файлов и средств их создания и отладки – **Редактора программного кода**. **M-**файлы представляют собой текстовые файлы, которые могут храниться в файлах («библиотеках») Matlab с расширением **m**.

Если вспомнить технологию процедурного программирования [x], то **m**-файлы фактически являются процедурами системы Matlab. Эти **m**-файлы могут состоять из следующих **элементов (средств языка программирования Matlab)**:

- описания данных различного типа;
- описания констант и переменных, в том числе системных;
- операции;
- системные команды и функции;
- функции пользователя;
- оператор присваивания и управляющие операторы;
- системные операторы и функции;
- средства работы с файлами данных;
- средства расширения языка.

Обратите внимание, что каждому типу данных можно соотнести некоторые характерные для него операции, называемые методами (система Matlab является объектно-ориентированной системой). А поскольку в иерархии типов данных сверху находятся данные типа Array, это значит, что все виды данных в MatLab являются массивами.

Программирование инженерных задач в среде MatLab очень напоминает программирование на универсальных языках программирования. Однако поскольку в Matlab можно использовать не только выражения над структурированными данным (например, массивами) но и системные команды и функции, то система программирования Matlab является мощной программной системой.

При создании новых **m**-файлов Редактор можно открыть путем активизации инструментов **New Script** или **New** (рис. 1.1.1-1), а для редактирования существующих инструментом **Open** или двойным щелчком мышки по имени **m**-файла. При этом в **Рабочей среде** добавляются еще три вкладки инструментальной панели: **EDITOR, PUBLISH** и **VIEW** (рис. 1.3.1-1).

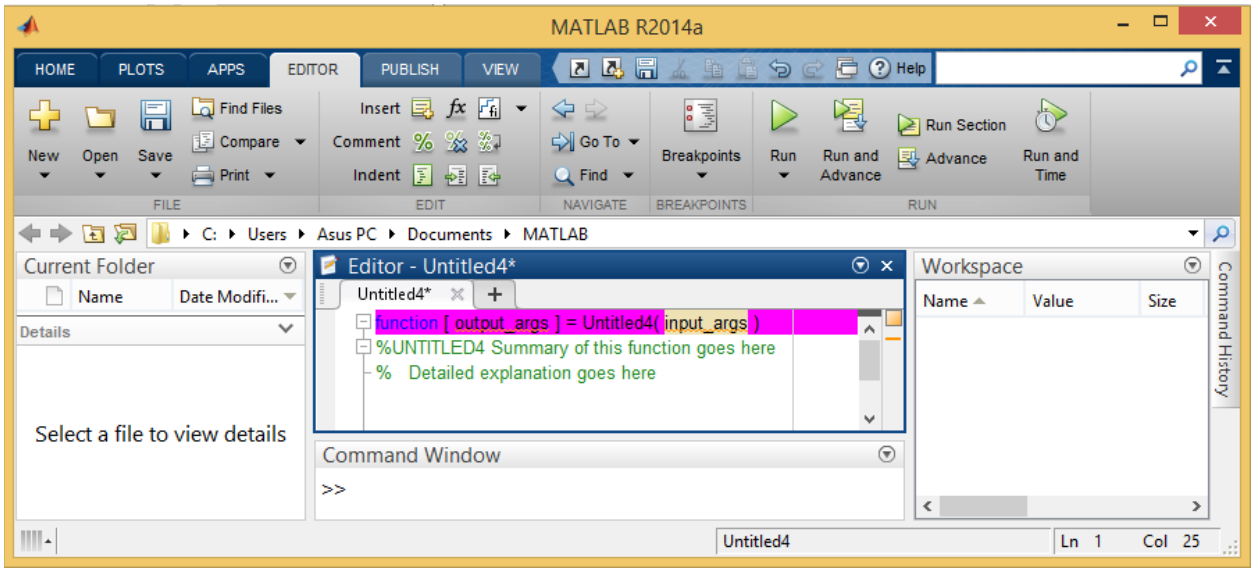


Рис. 1.3.1-1. Рабочая среда при открытиях **m**-файла

При активной вкладке **EDITOR** инструменты инструментальной панели позволяют открывать, сохранять, редактировать, запускать и осуществлять отладку **m**-файлов.

Эти инструменты разбиты на следующие категории:

- ***FILE*** – категория, включающая инструменты, которые позволяют создавать новые наборы команд и программы, и сохранять их в файлах; открывать существующие наборы команд и программ и загружать их из файлов; создавать различные объекты Matlab, осуществлять поиск файлов различных типов т.д.
- ***EDIT*** – категория, включающая инструменты, которые позволяют работать с текстом **m**-файла в окне редактора: вставлять в текст **m**-файлов различные элементы; превратить в комментарий текущую строку или вернуть ее к исходному виду; уменьшить или увеличить отступы текущей строки или нескольких выделенных строк на

заданное число позиций влево или вправо; выполнить интеллектуальный отступ.

- ***NAVIGATE*** – категория, включающая инструменты, которые позволяют осуществить навигацию в текущем **m**-файле: перейти к следующей или предыдущей строке; быстро перейти к строке с заданным номером и др. инструменты.
- ***BREAKPOINTS*** – категория, включающая инструменты, которые позволяют управлять точками останова во время отладки **m**-файла.
- ***RUN*** – категория, включающая инструменты, которые позволяют записать **m**-файл в текущий каталог и запустить его на выполнение, а также выполнять **m**-файл по секциями и управлять ими.

В Matlab существуют два типа **m**-файлов: **m**-сценарии и **m**-функции.

M-сценарий представляет собой последовательность команд и функции Matlab (**без входных и выходных параметров**), которые оперируют данными из **Рабочей области**, причем результаты выполнения ***m-сценария*** доступны **Рабочей области** и могут быть использованы для дальнейших вычислений. В среде Matlab **m-*сценарий*** принято также называть **script-*файлом*** или просто **script**, поэтому в дальнейшем мы будет использовать этот термин.

M-функции – это функции Matlab, аналогичные функциям языков программирования высокого уровня, таких как C++,C# и VB, которые могут иметь как входные, так и выходные параметры, также локальные переменные.

1.3.2. Описание и работа со script-файлами

Script является простейшей реализацией **m**-файлов. Он может содержать последовательность команд, операторов, функций и комментарии. При создании нового **script** возникает следующее окно редактора (рис. 1.3.2-1).

Рис. 1.3.2-1. Структура **script-*файла***

Script-*файлы* имеют свои особенности:

- не имеют входных и выходных параметров;
- работают с данными из рабочей области;
- в процессе выполнения не компилируются;
- строки автоматически нумеруются;

- представляют собой зафиксированную в виде файла последовательность команд, операторов и функций, полностью аналогичную той, что используется во время сессии в **Command Window**.

Откроем окно **Editor** для создания **script** и введем в нем команды, например, для построения графика (рис. 1.3.2-2).

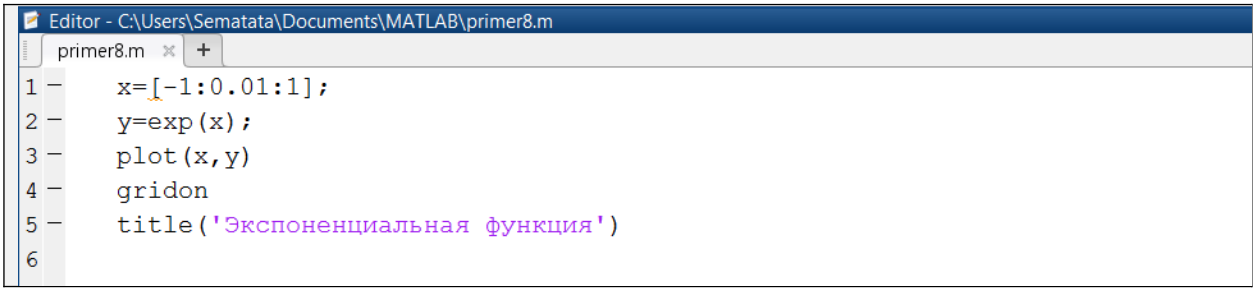


Рис. 1.3.2-2. **Script-файл**, содержащий команды для построения графика

Для сохранения созданного **script** следует щелкнуть по кнопке инструмента **Save** и в открывшемся меню выбрать команду **SaveAs**, в соответствующей строке окна ввести имя **m**-файла и щелкнуть по кнопке **Сохранить**. Имя файла появится в окне **Current Folder** с расширением **.m.**

Запуск **script** (рис. 1.3.2-3), сохраненного, например, с именем **пример9.m**, можно произвести из командной строки окна **Command Window,** введя его имя (без расширения), и нажать <Enter> (то есть выполнить, как команду **MatLab**). Зададим предварительно в командном окне переменным **x** и **z** числовые значения, а следом имя выполняемого файла **пример9**. Запуск файла позволил в следующей строке командного окна получить результат выполнения (рис. 1.3.2-4).

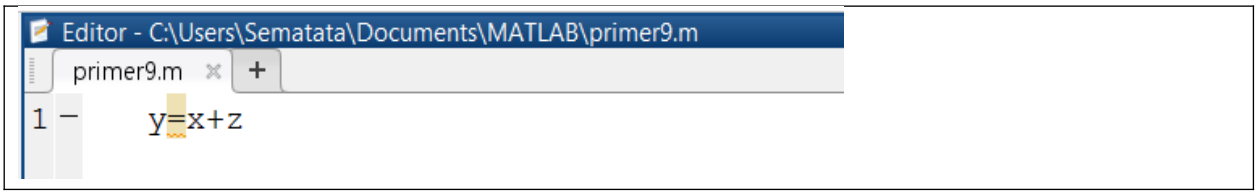


Рис. 1.3.2-3. **Script**-файл **primer9.m**

Рис. 1.3.2-4. Выполнение **script**-файла с именем **primer9**

Выполнение **script** или его части можно осуществить на этапе отладки (предварительно выполнив его сохранение). Для этого следует выделить нужные операторы и нажать инструмент **Run**. Выделенные операторы выполняются последовательно, точно так же, как если бы они были набраны в командной строке.

Рассмотрим еще один пример работы со **script** (**primer1**), который предназначен для построения графика (рис. 1.3.2-5).

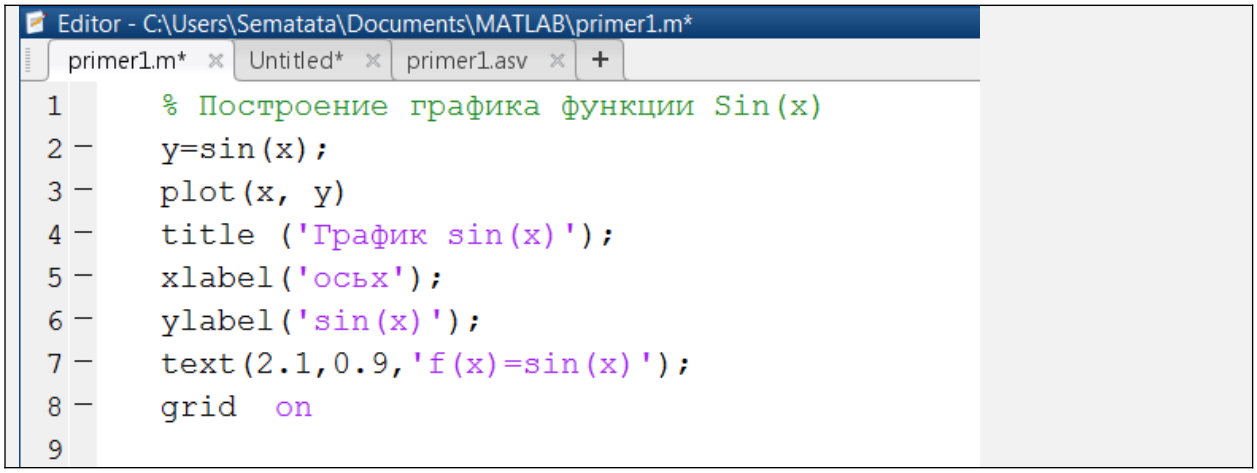


Рис. 1.3.2-5. **Script**-файл с именем **primer1**

Чтобы запустить этот файл на выполнение, следует предварительно задать последовательность значений переменной **x**, которая используется в теле файла (помня, что **script** работает с данными из рабочей области), и набрать в командной строке имя (рис. 1.3.2-6).

Рис. 1.3.2-6. Выполнение **script**-файла с именем **primer1**

Этот пример еще раз подтверждает, что все переменные, используемые в **script**, являются **глобальными**, т.е. они действуют одинаково в командах сессии и внутри программного блока, которым является **script**.

Результат выполнения команд **script** с именем **primer1** приведен на рис. 1.3.2-7.

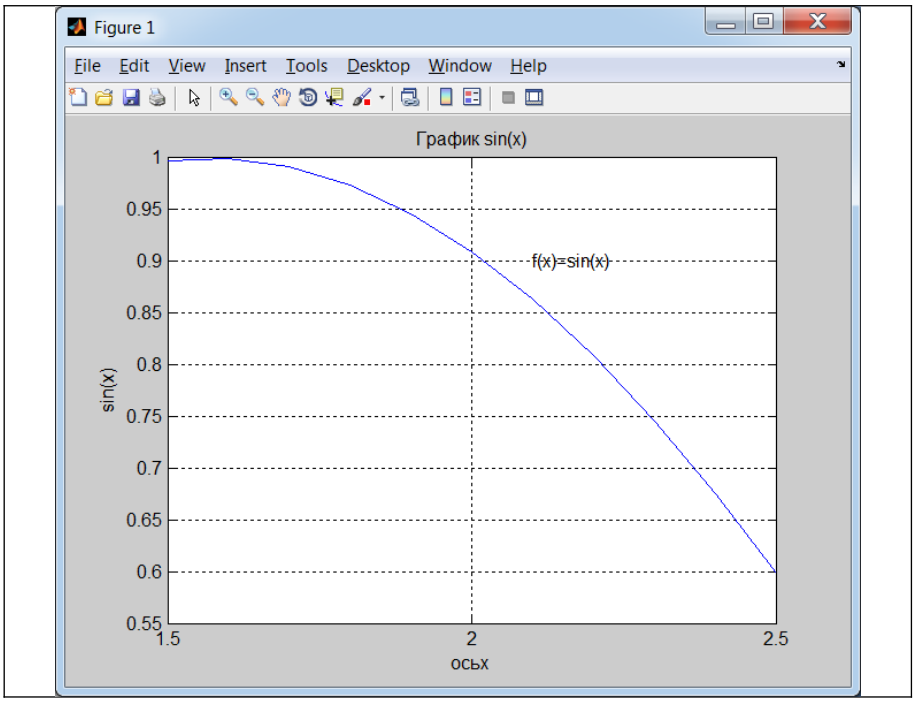


Рис. 1.3.2-7. Результат работы **script**-файла **primer1**

1.3.3. Описание и работа с m-функциями

M-функции, так же как и **script**, содержат команды, операторы и функции, но являются более сложным типом **m**-файлов по сравнению со **script** и имеют свои особенности:

- начинаются с заголовка описания **m**-функции;
- могут иметь входные и выходные параметры;
- все переменные, описанные в теле **m**-функции, являются **локальными**, т.е. действуют только в пределах тела функции;
- являются самостоятельными программными единицам, которые общаются с другими модулями посредством имени с входными и выходными параметрами.

В отличие от **script** **m**-функция является типичным объектом языка программирования высокого уровня.

При создании новой **m**-функции открывается окно редактора со следующим шаблоном (рис. 1.3.3-1).

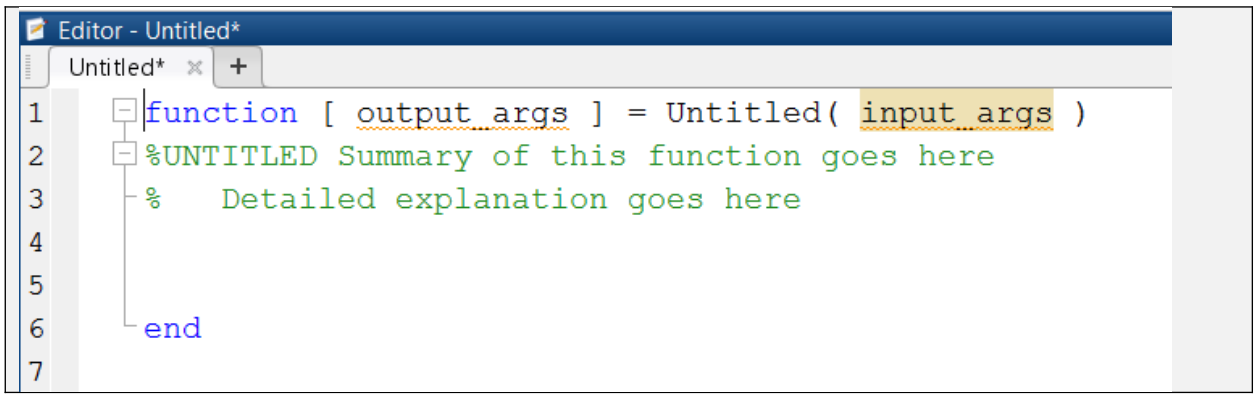


Рис. 1.3.3-1. Структура новой **m-функции**

Общая структура **m**-функции с **n** входными и **m** выходными параметрами имеет вид:

function[*var1, …, var**m**,…*] = *f_name (список входных параметров)*
% Основной комментарий
% Дополнительный комментарий
Тело m-функции
var1 = выражение
…
*var**m** = выражение*
end

Начинаются **m**-функции с заголовка **function**, затем в квадратных скобках через запятую указываются имена выходных параметров, далее ***f_name***–имя функции, а затем в круглых скобках - список входных параметров функции. Имена функций должны быть уникальными.

M-функция возвращает свое значение (или значения) и может быть вызвана из выражений, расположенных в рабочей области или в других программных модулях:

f_name(*список_параметров*)**.**

По умолчанию все переменные, описанные в теле **m**-функции, являются **локальными**, т.е. определены только в пределах функции, в которых они описаны. Между собой **m** -функции общаются посредством своего имени и входных и выходных параметров. Конструкция

vari=выражение,

приведенная в общей структуре **m**-функции, используется, если требуется, чтобы функция возвращала результаты вычислений.

Поскольку **m**-функция может иметь не один, а несколько выходных параметров, то она во многом напоминает процедуру. Поэтому ее нельзя использовать непосредственно в математических выражениях. Если функция, имеющая несколько выходных параметров, используется как функция, имеющая единственный выходной параметр, то для возврата значения будет использоваться первый из них. Это зачастую ведет к ошибкам в

математических вычислениях. Поэтому обращение к **m**-функции с несколькими выходными параметрами должно иметь вид:

[*var1,var2,...* **]=***f_name(Список_параметров)*

Создадим простейшую **m**-функцию с двумя входными и одним выходным аргументами. При активизации Редактора появился шаблон, который заполним следующей информацией (рис. 1.3.3-2).

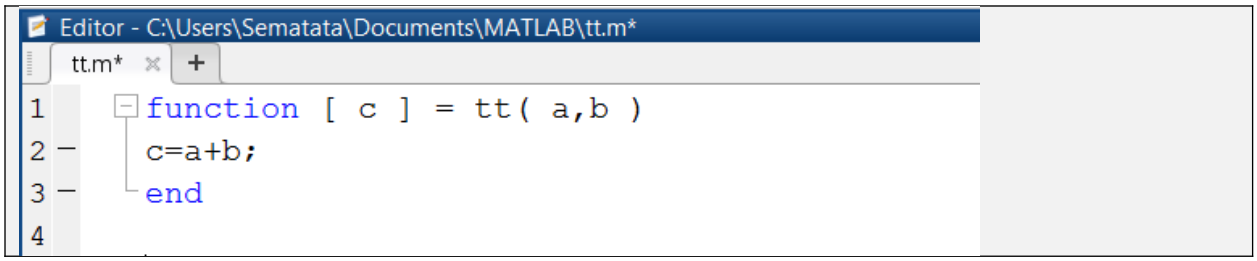


Рис. 1.3.3-2. Описание **m**-функции с одним выходным параметром

Сохраним функцию в файле, используя инструмент **Save**. При этом отметим, что **MatLab** предлагает в качестве имени **m**-файла название самой функции, т.е. **tt.m**. Всегда сохраняйте файл-функцию в **m**-файле, имя которого совпадает с именем m-функции! Теперь, убедившись, что каталог с файлом **tt.m**. является текущим, обратимся к функции **tt(2, 3)** из командной строки (рис. 1.3.3-3).

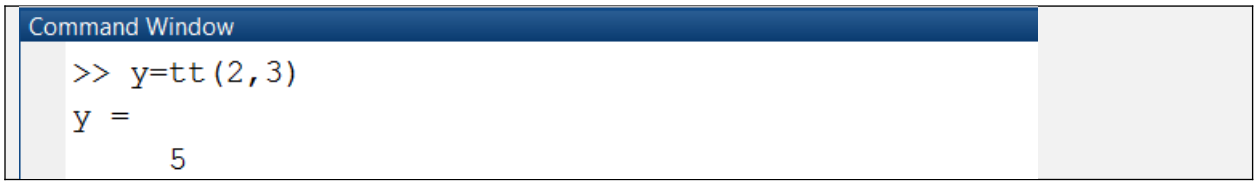


Рис. 1.3.3-3. Обращение к **m**-функции **tt(a,b)**

При вызове **m**-функции **tt(a, b)** входные аргументы **a** и **b** получили соответственно значения 2 и 3, сумма **a** и **b** записана в выходной параметр **c**, значение выходного аргумента **c** присвоено переменной **y**, а результат вывелся в следующую строку командного окна.

Следующий пример показывает создание **m**-функции **с несколькими выходными параметрами**. Список выходных параметров в заголовке **m**-функции заключен в квадратные скобки, а сами параметры отделены запятыми. В качестве примера создадим и выполним **m**-функцию **quadeq(a, b, c)**, которая по заданным коэффициентам квадратного уравнения находит его корни (рис. 1.3.3-4 , 1.3.3-5).

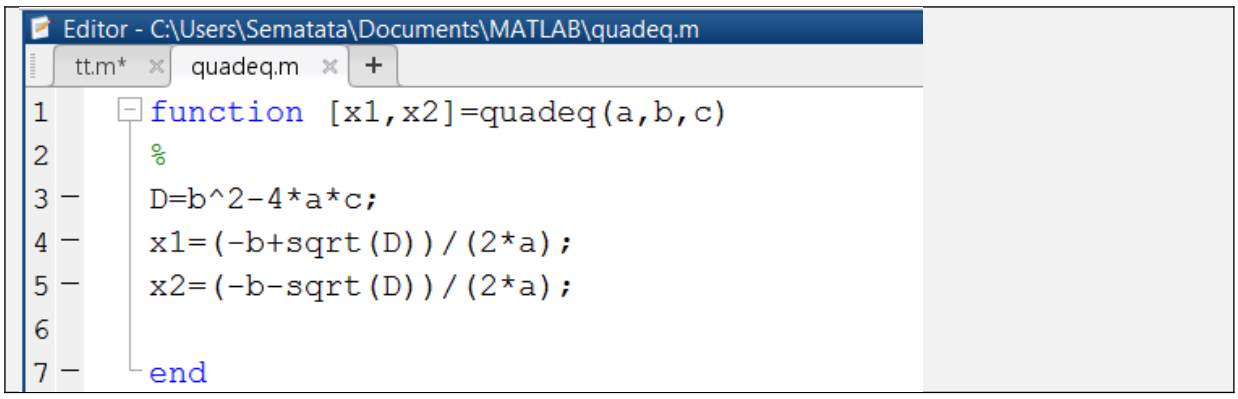


Рис. 1.3.3-4. Описание функции **quadeq(a, b, c)**
с двумя выходными параметрами

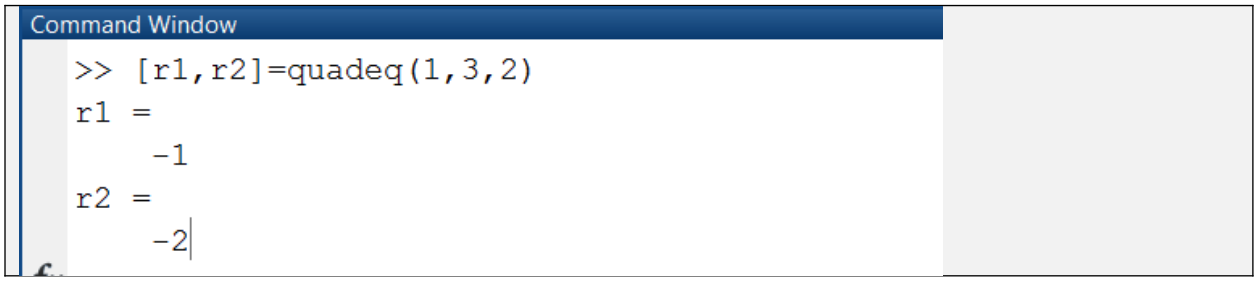


Рис. 1.3.3-5. Выполнение функции **quadeq(a, b, c)**
с двумя выходными параметрами

Иногда и при создании **m**-функций желательно применение **глобальных переменных** (например, если параметров слишком много). В таких случаях используемые глобальные переменные надо объявить командой:

global *varl, var2,...*

Для того чтобы несколько функций могли совместно использовать глобальные переменные, они должны быть объявлены как **global** в каждом из модулей.

1.3.4. Алгоритмические операторы Matlab

Помимо программ с **линейной структурой**, инструкции которых исполняются строго по порядку, существует множество алгоритмов, структура которых **нелинейная**. При этом последовательность элементов алгоритмов может выполняться в зависимости от определенных условий, иногда с конечным числом повторений – регулярных циклов, иногда в виде циклов, завершаемых при выполнении заданного условия. Практически любая серьезная программа имеет нелинейную структуру. Для создания таких программ необходимы специальные управляющие структуры. Они

имеются в любом языке программирования высокого уровня, и в частности в Matlab.

Рассмотрим операторы **m**-файлов подробнее.

Оператор присваивания. Основным оператором системы программирования **MatLab** является **оператор присваивания**, имеющий следующую структуру:

ИмяПеременной = *выражение*

Оператор предназначен для идентификации переменных и обозначается символом =, слева от которого находится имя переменной, а справа арифметическое или строковое выражение (правила записи арифметических и строковых выражений были рассмотрены в п. 1.1.2). Приведем несколько примеров операторов присваивания (рис. 1.3.4-1).

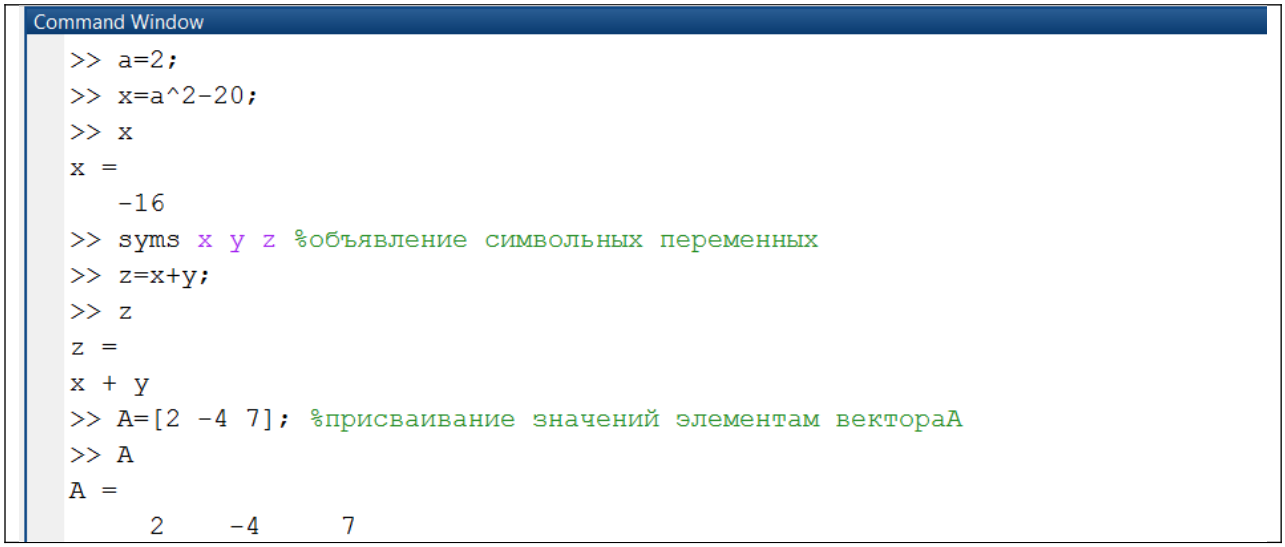


Рис. 1.3.4-1. Примеры операторов присваивания

Все переменные, используемые в правой части оператора присваивания, должны быть предварительно определены. Если командная строка заканчивается символом точка с запятой (**;**), то результат выполнения оператора не выводится, иначе он выводится в следующей строке командного окна. Это замечание распространяется и на выполнение операторов присваивания, расположенных в **m**-файлах.

Операторы ввода данных. Ввод данных в Matlab может осуществляться как с использованием оператора присваивания (**a=5**;), так и с использованием функции ввода данных с клавиатуры:

ИмяПеременной = **input ('***Запрос***');**

Эта функция вводит выражение с клавиатуры, а результат заносится в переменную с именем **a**. В приведенном ниже примере в переменную **a** введено вначале числовое значение, а затем числовое выражение (рис. 1.3.4-2).

```
Command Window
>> a=input('Введите значение переменной a=');
Введите значение переменной a=5
>> a=input('Введите значение переменной a=');
Введите значение переменной a=sin(3)
>>
```

Рис. 1.3.4-2. Ввод данных с клавиатуры

Функция **input()** может использоваться и для ввода произвольных строковых выражений. При этом она задается в следующем виде:

input('*Запрос***', V);**

При выполнении этой функции вычисления останавливаются в ожидании ввода строкового выражения. Введенное выражение выводится в следующей строке. Для вычисления выражения, заданного в символьном виде, использована функция **eval()**. Это иллюстрирует пример на рис. 1.3.4-3.

```
Command Window
>> S=input('Введите выражение ','S')
Введите выражение 2*sin(1)
S =
2*sin(1)
>> eval(S)
ans =
    1.6829
```

Рис. 1.3.4-3. Вычисление выражения, заданного в символьном виде

Условный оператор if…end. Условный оператор **if** в *общем виде* записывается следующим образом:

if *ЛогическоеВыражение1*
  Инструкции1
elself *Условие2*
  ЛогическоеВыражение2
else
  ЛогическоеВыражение3
end

Правила записи логических выражений описано в Теме 1.1.

Эта конструкция допускает несколько частных вариантов. Простейшее – **усеченное разветвление [x]** имеет следующий вид:

if *ЛогическоеВыражение*
Инструкции
end

Напомним, что если *ЛогическоеВыражение* возвращает логическое значение **1** (то есть «Истина»), выполняются *Инструкции*, составляющие тело структуры **if...end**. При этом оператор **end** указывает на конец перечня инструкций. Инструкции в списке разделяют запятая или точка с запятой. Если *ЛогическоеВыражение* не выполняется (дает логическое значение **0**, «Ложь»), то *Инструкции* также не выполняются.

Ниже приведен пример использования простейшего усеченного разветвления, реализованного с использования оператора **if** (рис. 1.3.4-4).

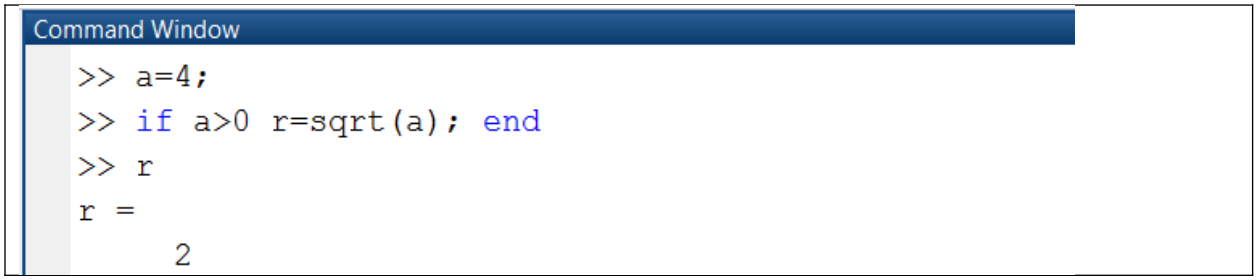


Рис. 1.3.4-4. Пример усеченного разветвления

Вторая частная конструкция напоминает **стандартное разветвление [x]**:

if *ЛогическоеВыражение*
Инструкции1
else
Инструкции2
end

Здесь выполняются *Инструкции1*, если выполняется *истинно ЛогическоеВыражение*, или, в противном случае, выполняются *Инструкции2*.

В примере, приведенном на рис. 1.3.4-5, рассматривается стандартное разветвление, реализованное с использованием оператора **if**.

```
Command Window
>> a=4;
>> if a>0 x=sqrt(a); else disp('Подкоренное выражение <0'); end
>> x
x =
     2
>> a=-4
a =
    -4
>> if a>0
x=sqrt(a)
else
disp('Подкоренное выражение <0')
end
Подкоренное выражение <0
```

Рис. 1.3.4-5. Пример стандартного разветвления

Из приведенного примера видно, что оператор **if** может быть как в одну строку, так и в несколько строк.

Рассмотрим пример более сложного - **вложенного разветвления**. Рассмотрим пример

$$t=\begin{cases}\max(x,y), & \text{если } xy<0;\\ \max(x^2,\sin(y),\cos(x)), & \text{если } xy>2;\\ \dots\end{cases}$$

причем, для того чтобы полностью отразить структуру сложного разветвления, не заботясь о переносе длинных командных строк, используем **m**-функцию (рис. 1.3.4-7). Подберем данные для проверки основного разветвления и обратимся к функции **raz()** с различными исходными данными (рис. 1.3.4-6).

```
Command Window
>> p=raz(-2,1)
p =
     1
>> p=raz(3,1)
p =
     9
>> p=raz(1,1)
p =
     1
```

Рис. 1.3.4-6. Обращение к функции **raz()** с различными исходными данными

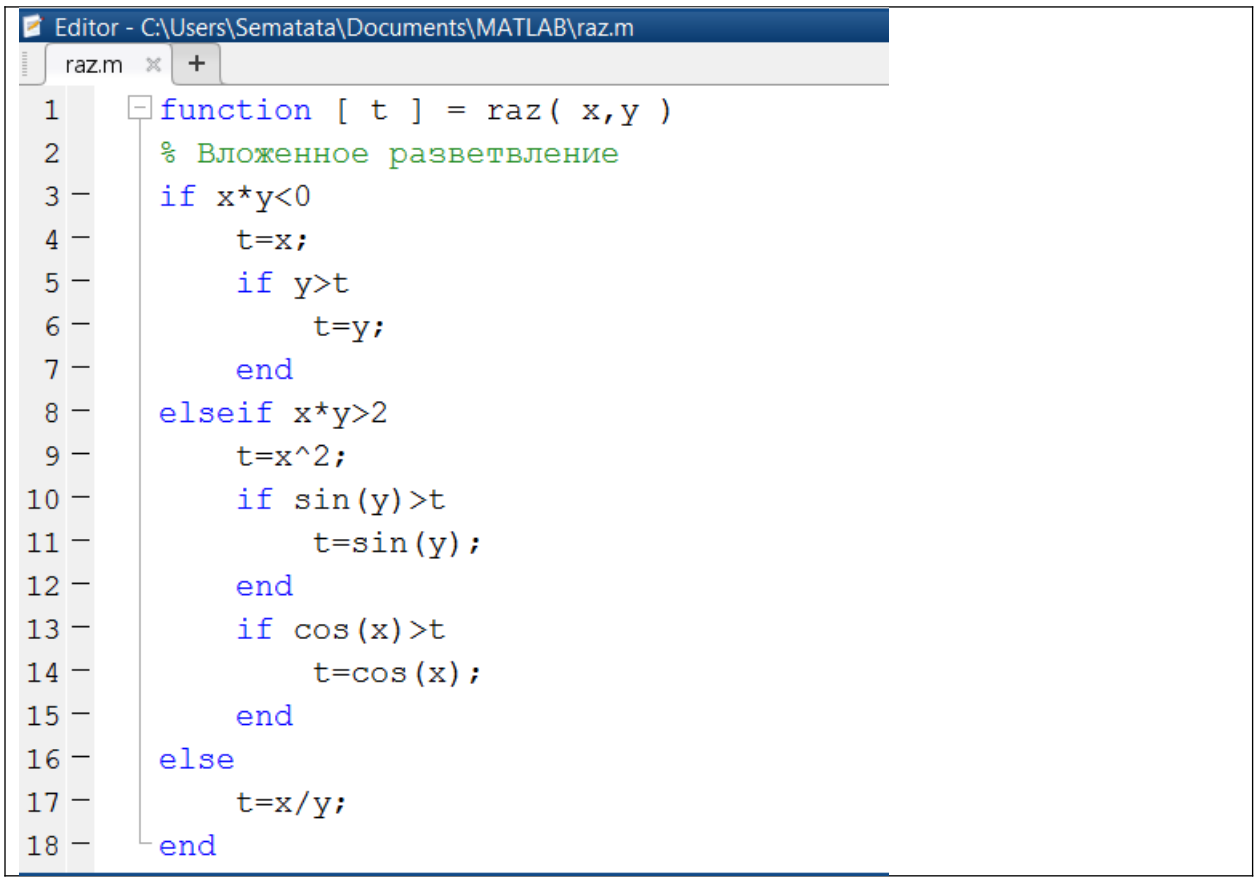


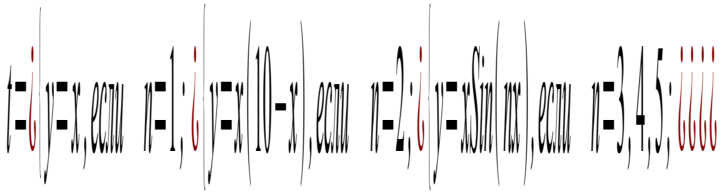
Рис. 1.3.4-7. Функция, реализующая вложенное разветвление

Оператор множественного выбора – switch. Для осуществления множественного выбора используется следующая конструкция **switch:**

switch *Выражение*
  case *Зачение_1*
    Список_инструкций_1
  case *Значение_2*
    Список_инструкций_2
  …
  case *Значение_N*
    Список_инструкций_N
  otherwise
    Список_инструкций_N+1
end

Если выражение после заголовка **switch** имеет значение одного из выражений *Значение...,* то выполняется блок операторов **case**, в противном случае — список инструкций после оператора **otherwise**. При выполнении блока **case** исполняются те списки инструкций, для которых *Значение* совпадает с *Выражением*. Обратите внимание на то, что *Значение* может быть числом, константой, переменной, вектором ячеек или даже строчной

переменной. Поясним использования оператора перебора **switch** следующим примером:



M-функция, реализующая множественное разветвление, приведена на рис. 1.3.4-8, а обращение к ней при исходных данных, позволяющих проверить каждую ветвь разветвления, показано на рис. 1.3.4-9.

Editor - C:\Users\Sematata\Documents\MATLAB\multifunc.m

multifunc.m

```
function [ y ] = multifunc( x,n )
switch n
    case 1
        y=x;
    case 2
        y=x*(10-x);
    case {3,4,5}
        y=x*sin(n*x);
    otherwise
        y=1/(1+x^2);
end
end
```

Рис. 1.3.4-8. Функция, реализующая множественное разветвление

Command Window

```
>> multifunc(5,1)
ans =
     5
>> multifunc(5,2)
ans =
    25
>> multifunc(5,4)
ans =
    4.5647
>> multifunc(5,-1)
ans =
    0.0385
```

Рис. 1.3.4-9. Обращения к функции **multifunc()**

У функции **multifunc(x,n)** два параметра, причем второй играет роль индикатора, определяющего тип функциональной зависимости. Значение функции записывается в переменную **y**. Если n=1, то выполняется первый case-блок, если 2, то – второй, если n=2, 3 или 4, то – третий. Если же значение переменной n не совпадает ни с одним из перечисленных значений, то выполняется команда, расположенная после ключевого слова **otherwise**.

Оператор регулярного цикла – for...end. Оператор цикла типа **for...end** обычно используется для организации вычислений с заданным числом повторений циклов. Конструкция такого цикла имеет следующий вид:

```
for    var = s:d:e
       Инструкция1
       ....
       ИнструкцияN
end
```

где **s** - начальное значение переменной цикла **var**, **d** - приращение этой переменной и **e** - конечное значение управляющей переменной, при превышении которого цикл завершается. Возможна и запись в виде **s:e** (в этом случае **d=l**). Список выполняемых в цикле инструкций завершается оператором **end**.

В качестве примера использования оператора **for...end** вычислим сумму элементов массива **x**, значения которого определены в командном окне с использованием m-функции **summa()** (рис. 1.3.4-10), параметром которой служит вектор **x**. Количество элементов массива **x** определяется функцией **length**. Кроме обращения к функции в командном окне предусмотрена проверка результата вычислений с использованием встроенной функции **sum(x)** (рис. 1.3.4-11).

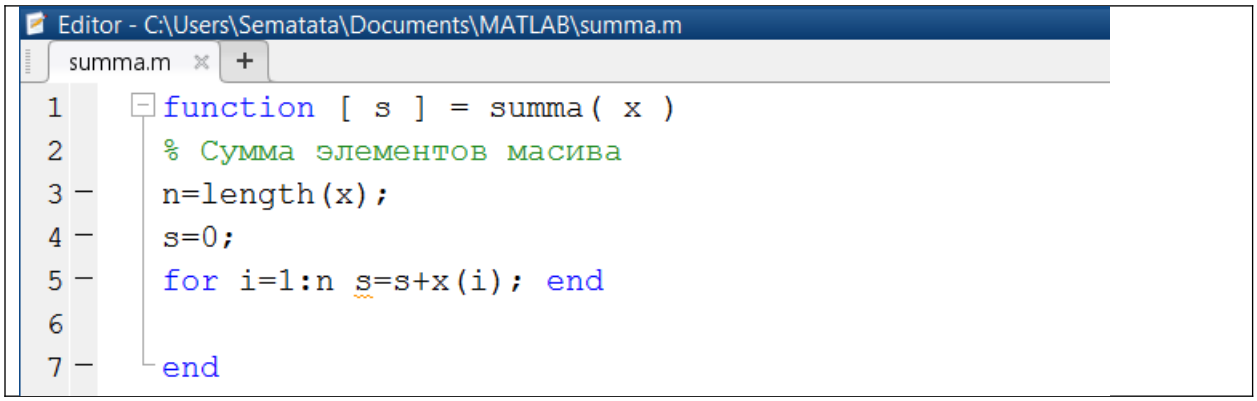


Рис. 1.3.4-10. Функция, вычисляющая сумму элементов массива

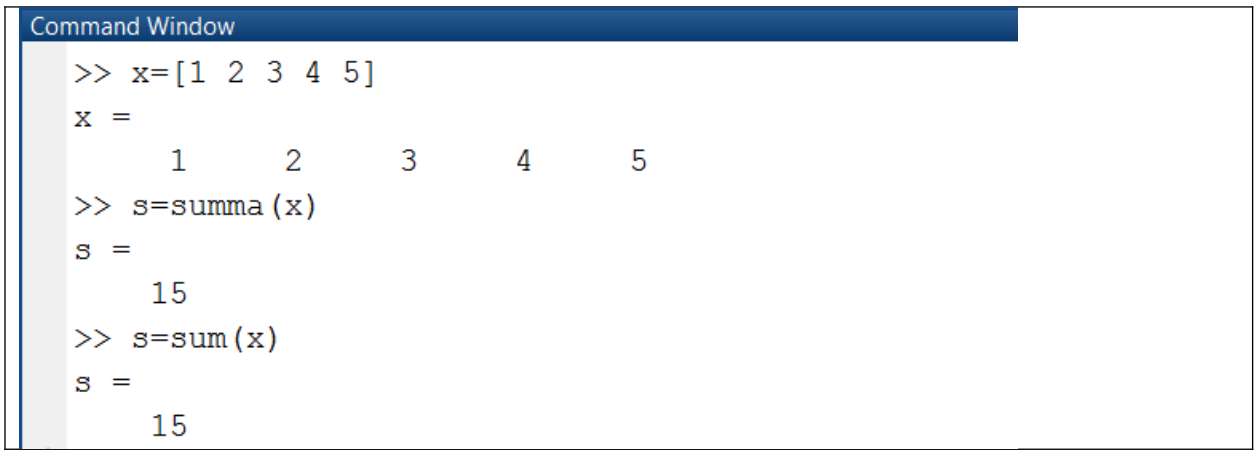


Рис. 1.3.4-11. Обращение к функции **summa()** и встроенной функции **sum()**

В цикле может быть использован оператор ***continue***, который передает управление в следующую итерацию цикла, пропуская операторы, которые записаны за ним, причем во вложенном цикле он передает управление на следующую итерацию основного цикла. Оператор ***break*** может использоваться для досрочного прерывания выполнения цикла (например, при отладке участка программы). Как только он встречается в программе, цикл прерывается.

Кроме простых регулярных циклов в Matlab имеется возможность организации **вложенных циклов**. Рассмотрим пример формирования двумерного массива **a,** каждый элемент которого представляет сумму его индексов (рис. 1.3.4-12). Обращение к **script**-файлу **vzikl** приведено на рис. 1.3.4-13.

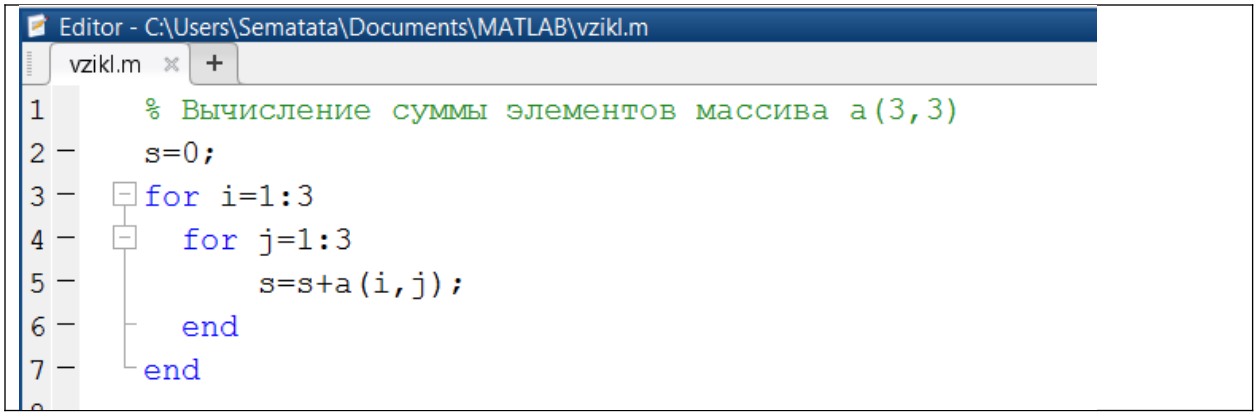


Рис. 1.3.4-12. **Script**-файл, иллюстрирующий вложенные циклы

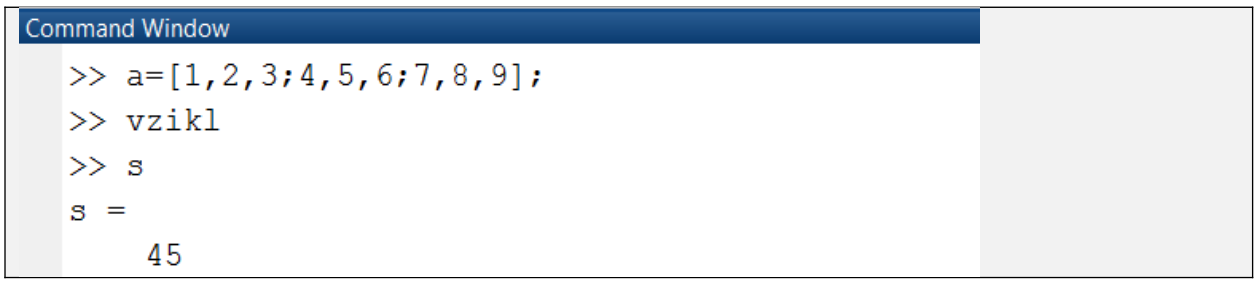


Рис. 1.3.4-13. Обращение к **script**-файлу с именем **vzikl**

Оператор итеративного цикла – while…end. Общий вид структуры ***while…end*** выглядит следующим образом:

while *ЛогическоеВыражение*
Инструкции
end

Отличительной особенностью этой структуры является то, что инструкции, расположенные в теле структуры повторения, выполняются только в том случае, если некоторое *ЛогическоеВыражение* «истинно». Как только условие становится «ложным», происходит выход из структуры повторения, и управление передается на инструкцию, расположенную после ключевого слова **end**.

Приведем простой пример (рис. 1.3.4-14).

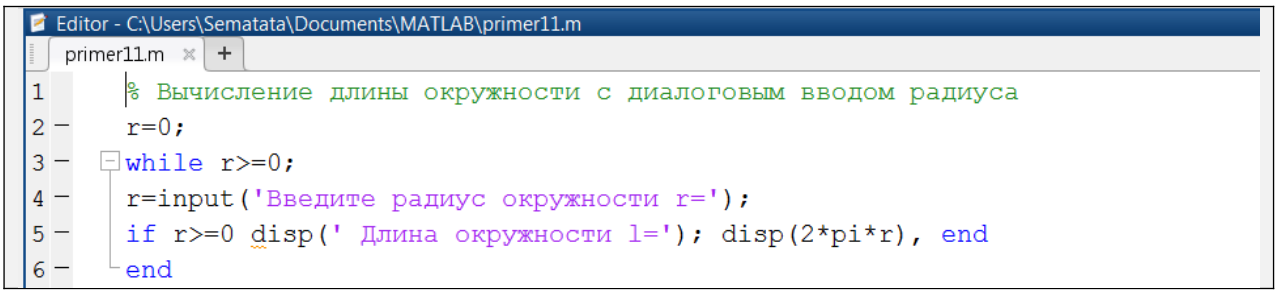


Рис. 1.3.4-14. Диалоговая программа, использующая оператор ***while…end***

Эта программа, сохраненная в **m**-файле с именем **primer11**, служит для многократного вычисления длины окружности по вводимому пользователем значению радиуса **r**, где диалог реализован с помощью команды **input.** Строки, связанные с вводом переменной **r** и вычислением длины окружности, включены в управляющую структуру **while...end**. Это необходимо для циклического повторения вычислений при вводе различных значений **r**. Пока **r>=0**, цикл повторяется. Но стоит задать **r<0**, вычисление длины окружности перестает выполняться, а цикл завершается. Поскольку во второй строке программы величина **r** определена равной 0, цикл повторяется хотя бы один раз.

Работа с программой в командном окне представлена на рис. 1.3.4-15.

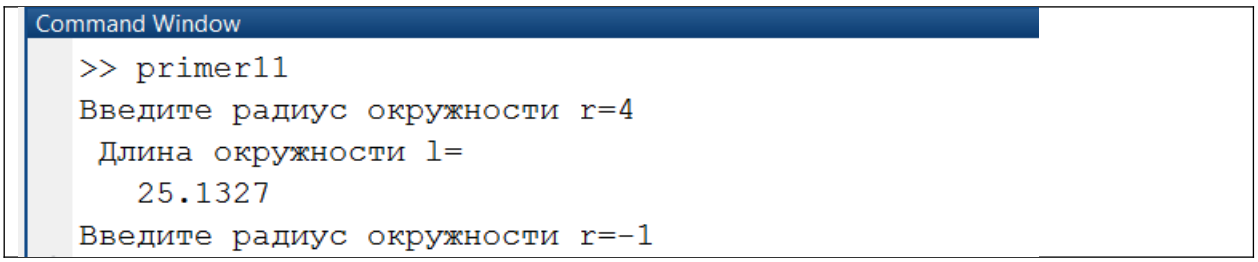


Рис. 1.3.4-15. Обращения к программе вычисления длины окружности

В управляющих структурах, в частности в циклах **for** и **while**, часто используются операторы, влияющие на их выполнение. Так, оператор **break**

может использоваться для досрочного прерывания выполнения цикла. Как только он встречается в программе, цикл прерывается.

Рассмотрим пример досрочного прерывания цикла при выполнении условия i=5 (рис. 1.3.4-16).

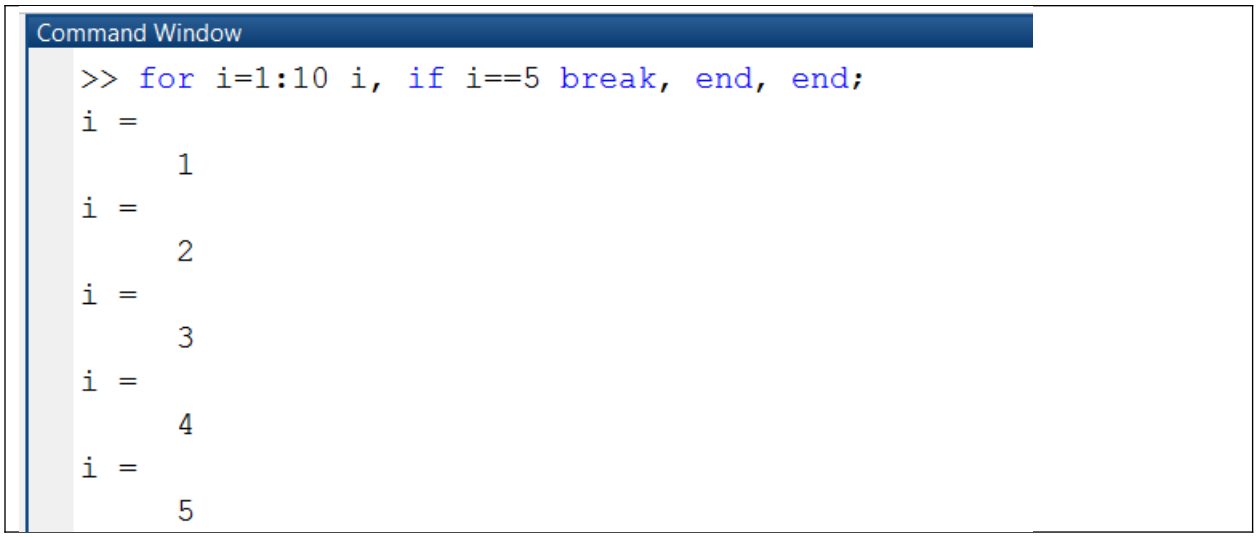


Рис. 1.3.4-16. Прерывание программы с применением оператора **break**

Оператор **continue** передает управление в следующую итерацию цикла, пропуская операторы, которые записаны за ним, причем во вложенном цикле он передает управление на следующую итерацию основного цикла. Ниже приведен пример вычисления суммы и произведения положительных элементов двумерного массива b(3,3) (рис. 1.3.4-17).

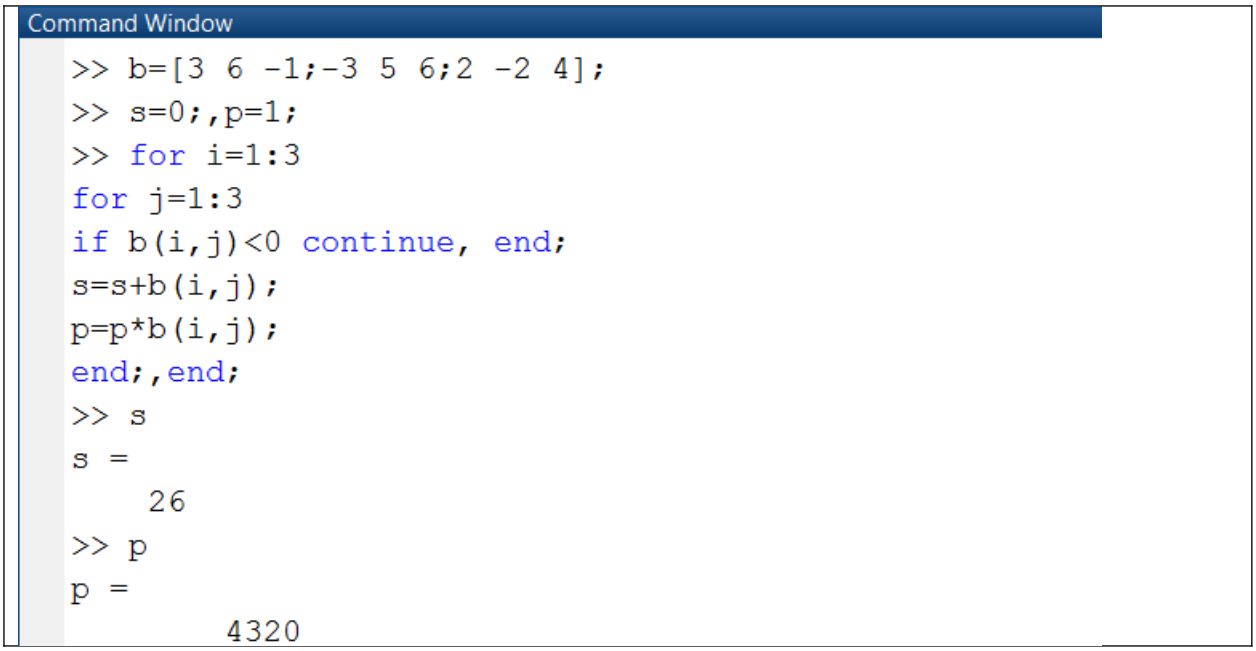


Рис. 1.3.4-17. Прерывание программы с применением оператора **continue**

1.3.5. Примеры решения задач с использованием m-файлов

Пример 1.3.5-1. Даны n чисел $b_1, b_2, \ldots b_n$. **Требуется вычислить их сумму:** $f(b_1)+f(b_2)+\ldots+f(b_n)$, *где*

$$f(x)=\begin{cases} x^2, & \text{если } x \text{ кратно } 3-\text{м}; \\ x, & \text{если } x \text{ при делении на } 3 \text{ дает остаток } 1; \\ \ldots & \end{cases}$$

Для решения поставленной задачи разработана функция **fb(x)**, реализующая алгоритм вычисления текущего значения функции. Функция имеет один входной параметр – текущее значение элемента массива **b** и один выходной параметр – **y** (рис. 1.3.5-1). Обращение к функции происходит в цикле, организованном для вычисления суммы (рис. 1.3.5-2).

Editor - C:\Users\Sematata\Documents\MATLAB\fb.m

zadasha.m | fb.m | +

```
1   function [ y ] = fb( x )
2   %вычисление значения функции f(x) по
3   % алгоритму разветвления
4   if fix(x/3)==x/3
5       y=x^2;
6   elseif mod(x,3)==1
7       y=x;
8   else
9       y=x/3;
10  end
```

Рис. 1.3.5-1. Функция, реализующая алгоритм Примера 1.3.5-1

Editor - C:\Users\Sematata\Documents\MATLAB\zadasha.m

zadasha.m | fb.m | +

```
1  n=10;
2  s=0;
3  b=[3 5 4 6 9 7 9 6 10 12];
4  for i=1:10
5      s=s+fb(b(i));
6  end
7  s
8
```

Рис. 1.3.5-2. Программа, реализующая вычисление суммы чисел

Для вычисления суммы значений функции создан **script**-файл с именем **zadasha.m**, в котором сначала заданы количество чисел (**n=10**) и вектор их значений (**b**), а затем организован регулярный цикл для обращения в функции **fb()** и вычисления суммы.

Вычисления производятся запуском **script**-файла путем набора в командной строке окна **Command Window** его имени **zadasha**. Результаты его выполнения выведены на рис. 1.3.5-3.

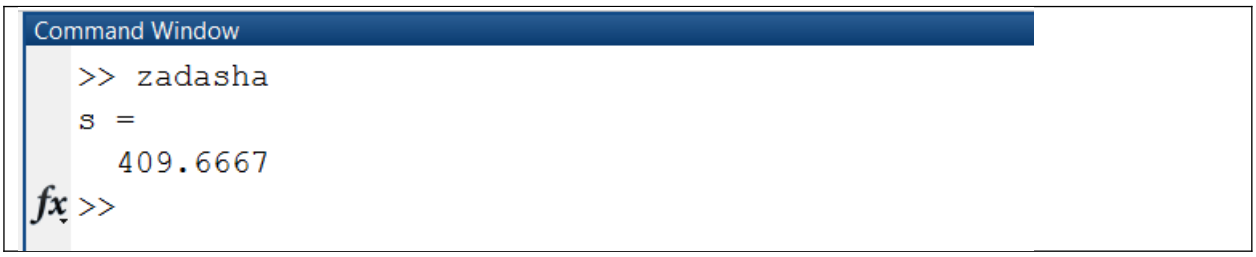


Рис. 1.3.5-3. Запуск **script**-файла **zadasha** на выполнение

Пример 1.3.5-2. Сформировать из произвольных чисел двумерный массив a(3,4). Вычислить и вывести одномерный массив b, каждый элемент которого есть среднее арифметическое элементов соответствующей строки массива a[].

На рис. 1.3.5-4 приведен **script**-файл с именем **zadasha2**, где введена матрица, **a**, состоящая из трех строк и четырех столбцов. Организован цикл по количеству формируемых элементов массива **b** путем обращения к функции **sred_ar()**. В функцию передается массив **a**, номер строки (**i**) и количество элементов в строке (**m**). Вывод элементов массива **b** предусмотрен в столбец.

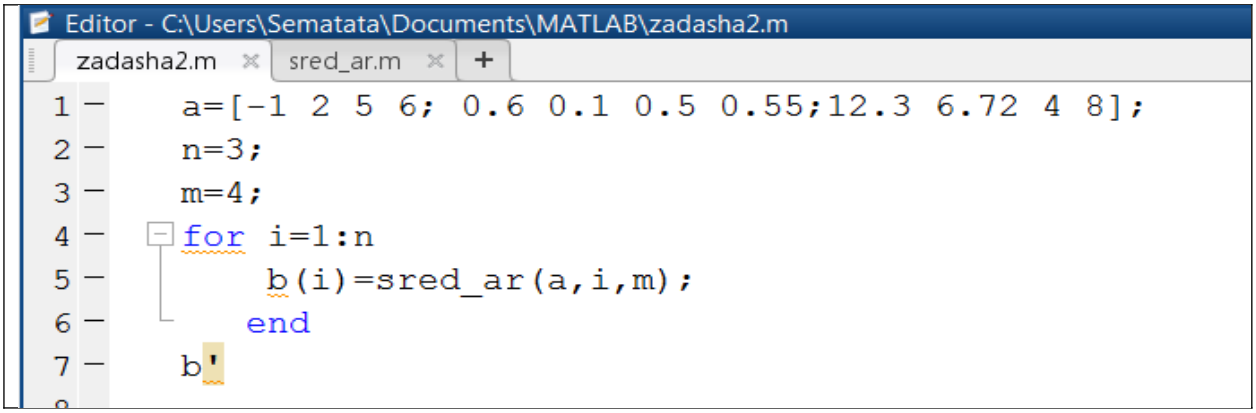


Рис. 1.3.5-4. Программа формирования массива **b**

Функция **sred_ar()** (рис. 1.3.5-5) предназначена для формирования **i**-го элемента массива **b**, равного среднему арифметическому элементов строки массива a.

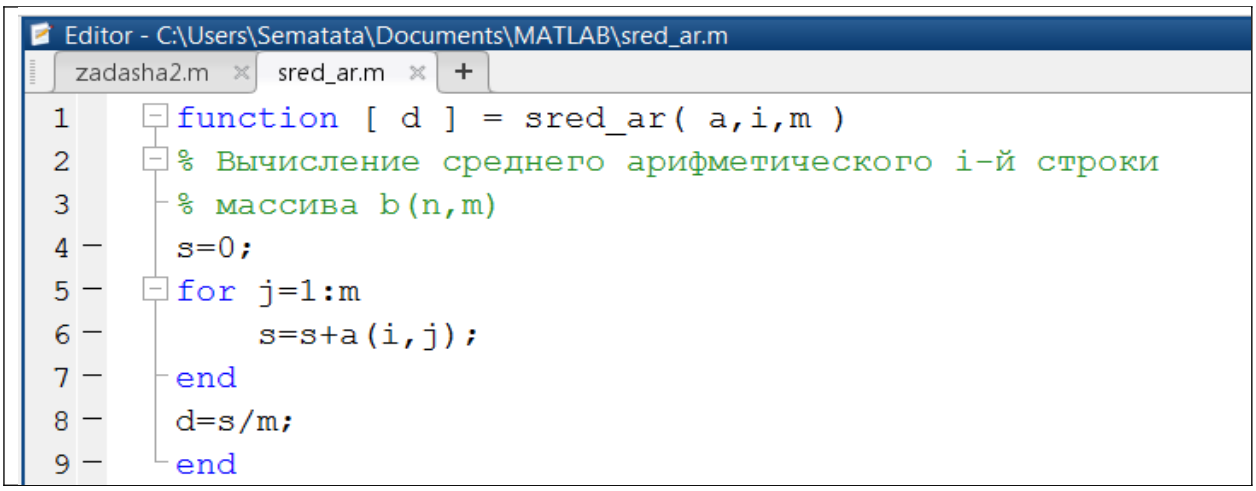


Рис. 1.3.5-5. Функция **sred_ar()**, вычисляющая среднее арифметическое элементов строки массива **a**

В результате запуска **script**-файле с именем **zadasha2** в окно **Command Window** выводится столбец элементов массива **b**

Рис. 1.3.5-6. Запуск **script**-файла **zadasha2**

Пример. 1.3.5-3. Задать действительные числа a,b, натуральное n (a<b) и вычислить выражение $s=(F_1+F_2+\ldots+F_n)h$, **где** $h=\frac{b-a}{n}$, **если** $F_i=-\frac{a+(i-1/2)h}{1+(i-1/2)h^2}, \quad i=1,2,\ldots,n.$

Решение задачи требует разработки двух функций: **fab(a, h, i)**, предназначенной для вычисления i-го слагаемого (рис. 1.3.5-7) и **sumf(a, h, n),** предназначенной для вычисления заданного выражения (рис. 1.3.5-8).

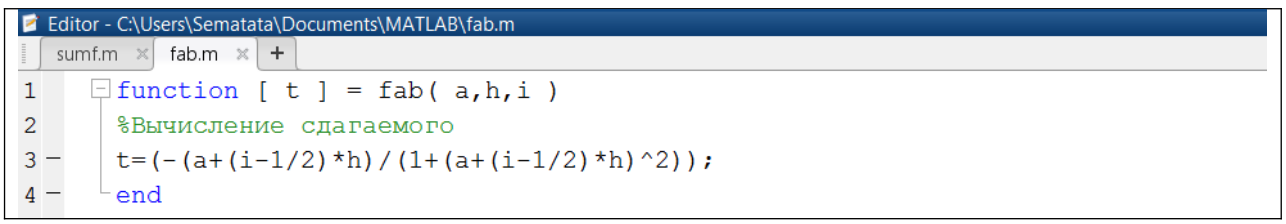


Рис. 1.3.5-7. Функция **fab(),** вычисляющая значение i-го слагаемого

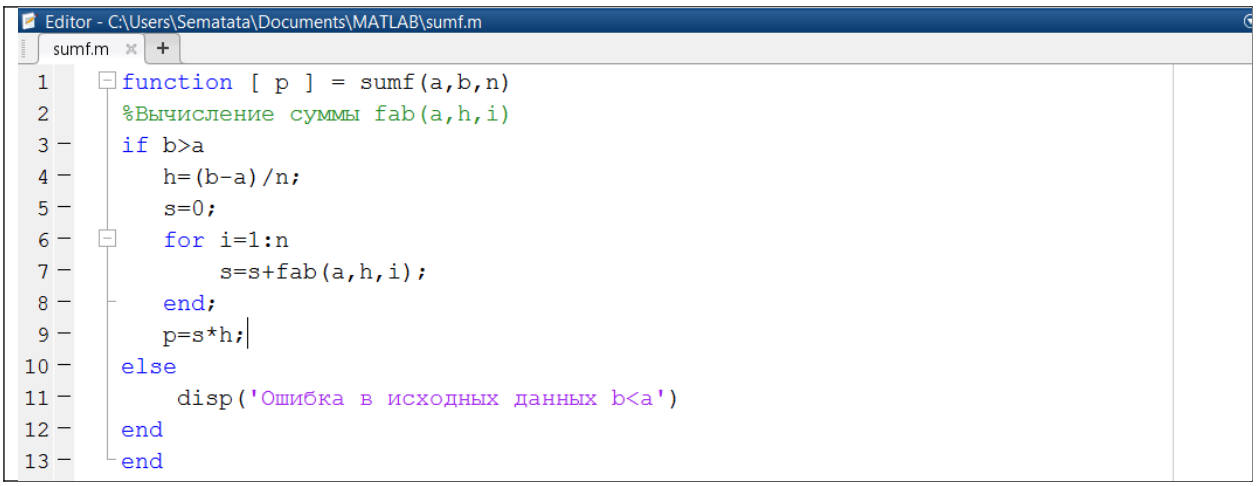


Рис. 1.3.5-8. Функция **sumf()**, вычисляющая заданное выражение

Запуск на выполнение осуществляется из командного окна к функции **sumf()**. Предварительно переменным **a**,**b** и **n** присваиваются числовые значения. Проверка правильности ввода исходных данных предусмотрена в функции **sumf()**. Вычисления выполняются, и результат выводится на экран только в случае, если b>a, иначе в командной строке появляется сообщение «Ошибка в исходных данных b<a» (рис. 1.3.5-9).

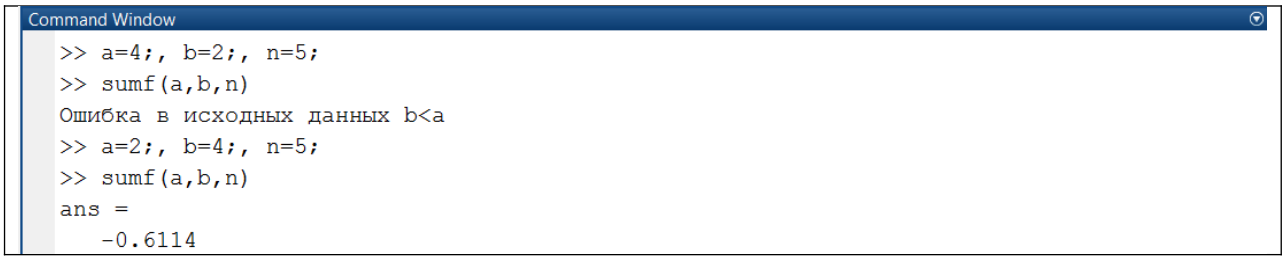


Рис. 1.3.5-9. Запуск функции **sumf()** на выполнение

1.3.6. Лабораторная работа по теме «Средства алгоритмизации и программирования в Matlab»

1. Вопросы, подлежащие изучению

1) Виды **m-**файлов.
2) Создание и сохранение новых, и открытие ранее созданных m-файлов.
3) Особенности **script-**файлов и **m-**функций.
4) Запуск на выполнение **script-**файла из текстового редактора.
5) Запуск на выполнение **script-**файла из командного окна.
6) Обращения к **script-**файлам и **m-ф**ункциям.
7) Средства языка программирования в системе Matlab.
8) Основные операторы m-языка их назначение и форматы**.**

2. Общее задание

1) ***Изучите материал Темы 1.3 (п.п. 1.3.1 – 1.3.5)***.
2) ***Выберите индивидуальное задание из табл. 1.3.6-1***.
3) ***Разработайте m****-функции для реализации стандартных алгоритмов: вычисления конечных сумм, разветвлений, поиска минимума и максимума в последовательности данных и т.п.*
4) ***Введите*** *и* ***сохраните m****-функции на внешнем носителе.*
5) ***Создайте*** *новый* ***script****-файл, в который введите код программы, описывающий логику решения поставленной задачи.*
6) ***Сохраните script****-файл в текущем каталоге.*
7) ***Произведите отладку script****-файла, запуская его на выполнение из текстового редактора командой* ***Run***.
8) ***Подготовьте*** *и* ***введите*** *исходные данные для решения поставленной задачи;*
9) ***Выполните script****-файл из командной строки окна* ***Command Window***.
10) ***Сохраните текст*** *рабочего окна на внешнем носителе.*
11) ***Предоставьте результаты*** *работы преподавателю,* ***ответьте*** *на поставленные вопросы.*
12) ***Выполните*** *команду* ***clear all*** *для очистки* ***Рабочей среды***.
13) ***Оформите отчет*** *по выполненной работе****.***

3. Варианты индивидуальных заданий

Таблица 1.3.6-1

№	Задание
1	Ввести натуральное число n*и вектор действительных чисел* $y_1, y_2 \dots y_n$ Найти: $\max(\lvert z_1\rvert, \dots \lvert z_n\rvert)$, где $z_i = \begin{cases} y_i, если & \lvert y_i\rvert \le 2; \\ 0.5, если & 2 < \lvert y_i\rvert < 4; \\ \dots \end{cases}$
2	Вычислить $\sum_{i=1}^{10} (a_i - b_i)^2$, где $a_i = \begin{cases} i, если & i - нечетное; \\ i/2, если & i - четное \end{cases}$ $b_i = \begin{cases} i^2, если & i - нечетное; \\ i^3/2, если & i - четное \end{cases}$
3	Задать массив $a_1, a_2 \dots a_{2n}$, состоящий из четного количества элементов. Каждая пара чисел $a_i + a_{i+1}$, где i+1 кратно двум, задает координаты вершины ломаной. Построить ломаную, соединив при этом последнюю вершину с первой
4	Ввести натуральное число n и вектор действительных чисел $b_1, b_2 \dots b_n$. Вычислить произведение $f(b_1) \cdot f(b_2) \dots f(b_n)$, где $f(x_i) = \begin{cases} x_i^2, если & x_i кратно\ 2; \\ 2x_i, если & x_i кратно\ 7; \\ \dots \end{cases}$
5	Ввести натуральное число n и действительное число x. $\sum_{i=0}^{n} \frac{1}{n!(n+i)!} \cdot \left(\frac{x}{2}\right)^{2i+n}$ Вычислить
6	Ввести натуральное число n. Найти наибольшее среди значений $k \cdot e^{Sin^2(k+1)}$, где k=1, 2,…,n, а также сумму всех полученных значений
7	Ввести натуральное число n. Среди значений $a_1, a_2 \dots a_n$, где $a_i = \frac{i-1}{i+1} + Sin\left(\frac{i-1}{i+1}\right)^2$ (i=1,2,…n), найти все положительные и вычислить их сумму
8	Ввести натуральное число n и вектор действительных чисел $b_1, b_2 \dots b_n$. Определить, положительных или отрицательных чисел в векторе больше, и определить наибольшее из отрицательных и наименьшее из положительных чисел

9	Ввести матрицу B(5,7) и сформировать из первых наибольших элементов строк вектор C(5). Вывести его элементы в строку и столбец
10	Сформировать вектор по правилу: $a_{k+1}=2a_k-a_{k-1}$, где k=2,3,…, 7, если $a_1=Cos^2(1), a_2=-Sin^2(1)$. Найти сумму квадратов тех чисел, которые не превосходят 2
11	Ввести натуральное число n и вектор действительных чисел $b_1, b_2 \ldots b_n$. Найти количество двух соседних положительных чисел и двух соседних чисел разного знака
12	Ввести квадратную матрицу A(4,4). Сформировать из максимальных элементов ее столбцов вектор X, вывести его элементы на экран в прямой и обратной последовательности
13	Ввести вектор целых чисел $b_1, b_2 \ldots b_{10}$. Преобразовать его таким образом, чтобы сначала располагались нули, затем все остальные элементы. Определить сумму и количество элементов, значения которых кратно 5
14	Ввести вектор вещественных чисел $z_1, z_2 \ldots z_{18}$. Создать из него массив x, каждый элемент которого максимальный из трех элементов, идущих подряд в массиве z
15	Сформировать матрицу A(4,4) по правилу: $A(i,j)=i+5j$, если $i\le 2$; Найти и вывести значения и индексы двух одинаковых элементов. Если таковых не окажется, вывести сообщение
16	Сформировать матрицу D(3,2) по правилу: $D(i,j)=\frac{i^2-j^2}{2.}$. Создать вектор из отрицательных элементов полученной матрицы
17	Задать натуральное число **n**. Посчитать, какая из матриц размером **n** на **n** содержит больше положительных элементов, если их элементы формируются по правилу: $a(i,j)=Sin(i+j/2)$; $b(i,j)=Cos(i^2+n)$; $c(i.j)=Sin\left(\frac{i^2-j^2}{n}\right)$. Вывести на экран сформированные матрицы
18	Ввести квадратную матрицу вещественных чисел A(4,4). Найти сумму наибольших значений элементов ее строк. Сформировать новую матрицу B(4,4) путем умножения каждого элемента матрицы A на найденную сумму и делением его на определитель исходной матрицы

19	Ввести матрицу вещественных чисел A(4,7) и получить из нее вектор C(4), элементы которого это: • наибольший из элементов в первой строке; • наименьший из элементов во второй строке; • среднее арифметическое элементов третьей строки; • сумма элементов четвертой строки
20	Ввести натуральное число n и матрицу вещественных чисел C(n,n). Найти среднее арифметическое наибольшего и наименьшего значений ее элементов и, заменив этим значением диагональные элементы, вывести матрицу C на экран
21	Ввести натуральные числа k1, k2 и действительную матрицу размера 8x4. Поменять в матрице местами элементы k1 и k2 строк
22	Ввести натуральное число n и матрицу вещественных чисел C(n,9). Найти среднее арифметического каждого из столбцов, имеющих четные номера
23	Ввести векторы действительных чисел x(5), y(6),z(7). Вычислить величину t по следующему алгоритму: $t=¿\left(\left(\max\left(x_1,x_2,\ldots x\right)+\min\left(y_1,y_2,\ldots y_6\right)\right)/2,\ \ если\ \ \left\lvert\min\left(z_1,z_2,\ldots z_7\right)\right\rvert<4;\ ¿¿¿¿$
24	Ввести векторы действительных чисел x(5). Получить для x=1, 3, 4 значения $p(x+1)-p(x),$ где $p(y)=a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1$
25	Ввести векторы действительных чисел x(10). Получить из него другой массив p(10), элементы которого упорядочены по возрастанию
26	Ввести матрицу вещественных чисел A(3,4). Заменить элементы строки матрицы с максимальной суммой значений элементов – единицами, с минимальной - двойками, а остальные элементы матрицы положить равными нулю
27	Сформировать матрицу A(4,4) по правилу $A(i,j)=5(i+j)-j.$ Удалить из него столбцы, содержащие элементы, меньшие 10
28	Сформировать матрицу B(9,3) по правилу $B(i,j)=Sin(i+j/2).$ Определить наименьший элемент в каждой строке матрицы и записать его в соответствующий элемент вектора C. Вывести полученный вектор C
29	Ввести матрицу вещественных чисел A(3,4), все элементы которой различны. В каждой строке следует выбрать наибольшее и наименьшее значения, а сумму индексов столбцов, в которых они расположены, записать в соответствующий элемент вектора C(3)
30	Ввести матрицу вещественных чисел A(4,4). Получить последовательности элементов главной и побочной диагонали, создать из этих элементов векторы B(4) и C(4) и вывести их на экран

4. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в окне **Command Window**, снабженный необходимыми комментариями.

1.3.7. Контрольные вопросы по теме

1) Что такое **script-**файл и каковы его особенности?
2) Каким образом **script-**файл запускается на выполнение?
3) Что такое **m-**функци**я**?
4) В чем отличие **script-**файла от **m-**функции?
5) Может ли m-**функция** иметь несколько выходных параметров?
6) Обращение к m-**функции**.
7) Формат оператора **input().**
8) Как с использованием оператора **if…end** реализовать стандартное, усеченное и вложенное разветвление?
9) Формат оператора множественного разветвления **switch**.
10) Формат оператора регулярного цикла **for…end**, особенности задания значений переменной цикла.
11) Назначение операторов **continue** и **brek**.
12) Оператор итеративного цикла **while…end** и его структура.

1.

Раздел 2. Технология решения вычислительных задач средствами MatLab

Тема 2.1. Решение нелинейных уравнений

2.1.1. Численное решение нелинейных уравнений
2.1.2. Лабораторная работа по теме
2.1.3. Контрольные вопросы по теме

2.1.1. Численное решение нелинейных уравнений

В общем виде уравнение можно представить, как **f(x) = 0**. В зависимости от вида функции **f(x)** различают алгебраические и трансцендентные уравнения.

Алгебраическими уравнениями называются уравнения, в которых значение функции f(x) представляет собой полином **n**-й степени (**n≥2**). Всякое неалгебраическое уравнение называется **трансцендентным**. Функция **f(x)** в таких уравнениях содержит хотя бы одну из следующих функций: показательную, логарифмическую, тригонометрическую или обратную тригонометрическую.

Решением уравнения **f(x)=0** называется совокупность корней $\bar{x}$, при которых уравнение обращается в тождество $f(\bar{x}) \equiv 0$. Однако, точные значения корней могут быть найдены аналитически только для некоторых типов уравнений. Еще меньше возможностей при получении точного решения трансцендентных уравнений. Следует отметить, что задача нахождения точных значений корней не всегда корректна. Так, если коэффициенты уравнения являются приближенными числами, точность вычисленных значений корней заведомо не может превышать точности исходных данных. Эти обстоятельства заставляют рассматривать возможность отыскания корней уравнения с ограниченной точностью (приближенных корней).

Задача нахождения корня уравнения с заданной точностью **ε** (**ε>0**) считается решенной, если найдено приближенное значение $\bar{x}$, которое отличается от точного значения корня ξ не более чем на значение ε:

$$|\xi - \bar{x}| \le \varepsilon.$$

Для отделения корней нелинейных уравнений применяются **графический** и **аналитический** методы. Для уточнения корней с заданной степенью точности существует множество численных методов. Самыми распространенными из них являются: **метод Ньютона**, **метод хорд**, **метод итераций** и **метод половинного деления** [3].

Процесс нахождения приближенного корня нелинейного уравнения состоит из двух этапов:

1) **отделение корня уравнения** (локализация корня на отрезке);
2) **уточнение корня с заданной точностью.**

Пример 2.1.1-1. Отделить и уточнить корень уравнения $2^x-4x=0$.

Согласно теореме о существовании и единственности корня на отрезке, найдем отрезок, на концах которого функция $f(x)=2^x-4x$ имеет разные знаки, а первая производная непрерывна и знакопостоянна (рис. 2.1.1-1).

```
Command Window
>> syms x;
>> f=2.^x-4.*x;
>> diff(f,x)
ans =
2^x*log(2) - 4
>> f=@(x)2.^x-4.*x;
>> f1=@(x)2.^x*log(2)-4;
>> x=0:0.2:1;
>> f(x)
ans =
    1.0000    0.3487   -0.2805   -0.8843   -1.4589   -2.0000
>> f1(x)
ans =
   -3.3069   -3.2038   -3.0854   -2.9494   -2.7932   -2.6137
```

Рис. 2.1.1-1. Отделение корня нелинейного уравнения

Условия существования и единственности корня на отрезке **[0;1]** выполняются. Команды построения графика функции **f(x)** на отрезке **[0;1]** и график функции приведены на рис. 2.1.1-2.

```
>> z=f(x);
>> plot(x,z,x,0*z,'k-')
>> title('2^x-4*x =0')
>> xlabel ('x')
>> ylabel ('f(x)')
```

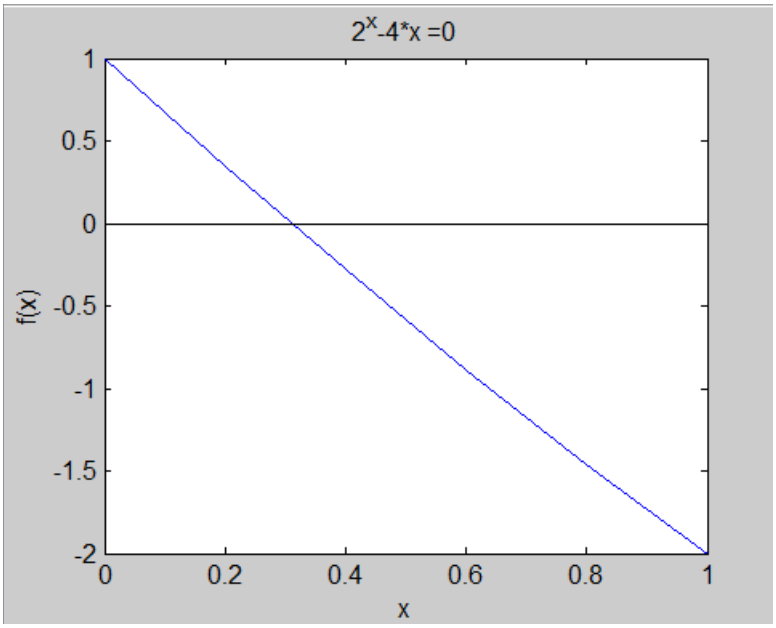


Рис. 2.1.1-2. Построение графика функции **f(x)**

Для решения нелинейных уравнений в среде Matlab используются встроенные функции: **solve(), fzero(), root()**.

Функция **fzero()** используется для нахождения вещественных корней уравнений вида **f(x)=0**. Алгоритм, реализованный этой функцией, представляет собой **комбинацию метода дихотомии (деления пополам), метода секущих и метода обратной квадратичной интерполяции**. В простейшем варианте обращения кроме указателя на функцию, корень которой необходимо найти, задается окрестность **x0**, с которой начинается поиск: **x = fzero(f, x0)**.

Аргумент **f** может быть задан одним из следующих способов:

- формулой с неизвестным **x**, заключенной в апострофы;
- именем **m-файла** (в апострофах и без расширения **m**);
- указателем на функцию (например, **@f_name**).

Следует отметить, что выражение, заключенное в апострофы (левая часть уравнения) в качестве независимой переменной может иметь переменную только с именем **x**. Использование независимой переменной с другим именем вызовет сообщение об ошибке.

Аргумент **x0** может быть задан одним из двух способов:

- вектором **[a;b]**, на концах которого функция **f(x)** меняет знак, что гарантирует нахождение, по крайней мере, одного корня на этом интервале;
- скалярным значением, в окрестности которого предполагается нахождение корня (в этом случае функция **fzero()** сама пытается найти отрезок с центром в заданной точке **x0**, на концах которого функция **f(x)** меняет знак).

Из графика (рис. 2.1.1-2) следует, что корень находится на интервале **[0;1]**. Используем полученную информацию и обратимся к функции Matlab **fzero()**. При этом если мы хотим получить не только значение корня, но и узнать значение функции в найденной точке, то к функции Matlab **fzero()** можно обратиться с двумя выходными параметрами. На рис. 2.1.1-3 рассмотрено обращение к функции **fzero()** как с одним, так и с двумя выходными параметрами.

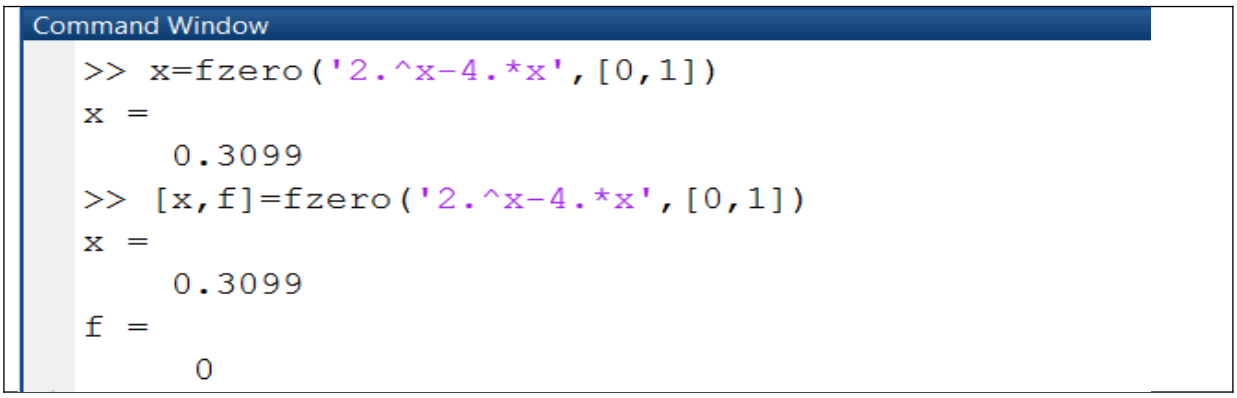


Рис. 2.1.1-3. Уточнение корня уравнения с использованием функции **fzero()**

Вместо явного задания выражения для функции **f(x)** можно создать соответствующую функцию, запомнив ее в виде **m**-файла (рис. 2.1.1-4).

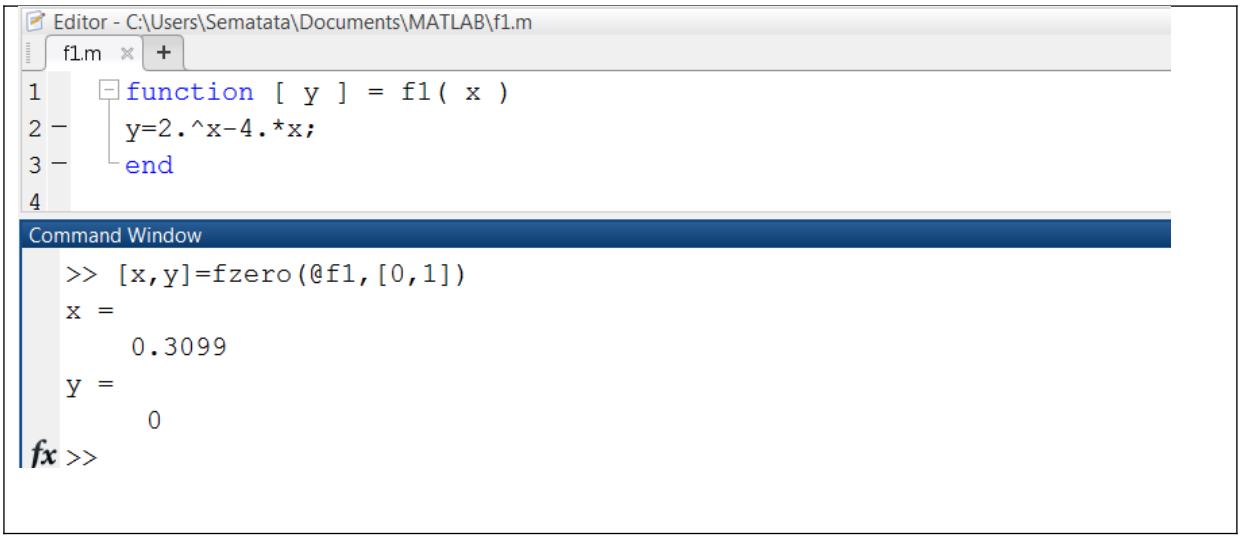


Рис. 2.1.1-4. Описание левой части в виде **m**-файла

Для символьного (аналитического) решения уравнений в MatLab используется функция **solve(),** которая может иметь следующие форматы:**solve('f(x)',x),solve('f(x)')**, где **'f(x)'**– левая часть решаемого уравнения, заключенная в апострофах; **x** – искомая символьная неизвестная (описанная как syms x).

Рассмотрим технологию определения корня с помощью функции **solve()** на следующем примере (рис. 2.1.1-5).

Пример 2.1.1-2. Решить уравнение 2^x–4·x+3=0 аналитически.

```
Command Window
>> syms x
>> y=solve('2*x-4*x+3=0')
y =
3/2
```

Рис. 2.1.1-5. Уточнение корня уравнения с использованием функции **solve()**

Функция **solve()** в ряде случаев позволяет определить все корни уравнения **f(x)=0**. Она не требует информации о начальном значении корня или области его изоляции. Поэтому в случае решения ряда нелинейных уравнений, она не находит всех корней уравнения.

Функция Matlab **roots()** используется для вычисления корней полинома вида $P(x)=a_1x^n+a_2x^{n-1}+\ldots+a_nx+a_{n+1}$. Перед ее использованием создается список коэффициентов (даже нулевых), а затем вводится сама функция **roots()**, у которой в качестве аргумента указывается имя списка коэффициентов (рис. 2.1.1-6).

```
Command Window
>> a=[1,-10,16,0,-1,10,-16];
>> roots(a)
ans =
   8.0000 + 0.0000i
   2.0000 + 0.0000i
  -1.0000 + 0.0000i
  -0.0000 + 1.0000i
  -0.0000 - 1.0000i
   1.0000 + 0.0000i
```

Рис. 2.1.1-6. Уточнение корня уравнения с использованием функции **roots()**

Вычисление корней полинома $P_6(x)=x^6-10x^5+16x^4-x^2+10x-16$ с использованием команды **roots(a)** показало, что у полинома есть четыре действительных корня: x_1**=-1,** x_2**=1,** x_3**=2,** x_4**=8** и два мнимых корня x_5**=i** и x_6**=-i**.

2.1.2. Лабораторная работа по теме «Технология решения нелинейных уравнений средствами пакета MatLab»

1. Вопросы, подлежащие изучению

1) Задание функций и их производных с использованием средств пакета Matlab.
2) Получение таблиц значений функций в заданных границах изменения аргумента.
3) Построение графиков функций средствами Matlab.
4) Этапы решения нелинейных уравнений: отделение и уточнение корня.
5) Решение нелинейных уравнений с использованием встроенных функций пакета Matlab: **fzero()** и **solve()**.

2. Общее задание

1) ***Изучите материал Темы 2.1 (п. 2.1.1)***.
2) ***Выберите индивидуальный вариант*** *задания из* ***табл. 2.1.2-1***.
3) ***Отделите корень*** *нелинейного уравнения* ***f(x)=0*** *с использованием средств пакета Matlab, для чего:*
   - ***Постройте графики*** *функции* ***f(x)*** *и ее первой производной;*
   - *на выбранном отрезке пересечения графика с осью* ***OX получить таблицы*** *значений функции* ***f(x)*** *и ее первой производной;*
   - ***проверить условие*** *существования единственного корня на выбранном отрезке;*
4) ***Решите нелинейное уравнение*** *с использованием функций* ***fzero()*** *и* ***solve();***
5) ***Предъявите*** *результаты выполнение задания на ПК преподавателю.*

3. Варианты индивидуальных заданий

Таблица 2.1.2-1

№	Уравнение	№	Уравнение
1	$x^2-6\ln(x+1)+2,8=0$	16	$2x^2-\sin x-1,5=0$
2	$x^2-4\sin x+0,8=0$	17	$2\cos(x-x^3)-x=0$
3	$x^2-3\sin x-2,4$	18	$x^2-3,4\ln(1+x)-1=0$
4	$x^2+\sin x-0,5\sqrt[3]{(1+2x)}=0$	19	$e^x-(x+1)^2-2=0$
5	$3x^2-2\cos(x+1)-4,8=0$	20	$x^2-\sin x=0$
6	$x^2-1,5\cos(x+3)-1,8=0$	21	$e^x-4x-9,5=0$
7	$\ln(x+2)+\cos(2+x)-0,3=0$	22	$x^2-3\sin x-2x=0$
8	$x^3-4,8\sin(1+2x)-1=0$	23	$3-2\sin x-\sqrt{x^2+1}=0$
9	$x^2-2\sin(x+1)-2=0$	24	$-2+3\sin x-\sqrt{x^2+2}=0$
10	$e^{x-3}-\ln(x+3)+1=0$	25	$e^{-0,4x}+3x^2-5x-3=0$
11	$-\sin x-\sqrt{x}+1=0$	26	$x^2+\cos(x-4)+\sqrt{x+1}-3=0$
12	$\cos(x-0,2x^2)-x+6=0$	27	$x^2-\ln x-3=0$
13	$e^x-4x=0$	28	$-x+e^{0,5x}=0$
14	$0,1x^2+x\ln x-x+0,5=0$	29	$-\sin x-\sqrt{1+x}+2=0$
15	$2e^{-x}+e^x-4=0$	30	$4x^3+2x^4+9,5e^{-2x}-35=0$

4. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в окне **Command Window**, снабженный необходимыми комментариями.

2.1.3. Контрольные вопросы по теме

1) Что называется, нелинейным уравнением?
2) Этапы решения нелинейного уравнения.
3) Графическое отделение корней нелинейного уравнения средствами Matlab.
4) Аналитическое отделение корней нелинейного уравнения средствами Matlab.
5) Способы задания функции нелинейного уравнения.
6) Назначение и формат функции **fzero().**
7) Назначение и формат функции **solve().**
8) Назначение и формат функции **roots().**

Тема 2.2. Технология аппроксимации интерполяции функций в среде пакета MatLab

2.2.1. Аппроксимация и интерполяция функций
2.2.2. Лабораторная работа по теме «Технология аппроксимации

интерполяции функций в среде пакета MatLab»

2.2.3. Контрольные вопросы по теме

2.2.1. Аппроксимация и интерполяция функций

Пусть имеется набор узловых точек $\mathbf{x_k}$ (где $\mathbf{k=1,2,\ldots,n}$) и значения функции $\mathbf{y(x_k)}$ в этих точках, а также некоторая функция $\mathbf{f(x,a_1, a_2, \ldots,a_m)}$, которая кроме аргумента **x** зависит еще и от параметров $\mathbf{a_s}$ (где $\mathbf{s=1, 2, \ldots m}$). Задача аппроксимации состоит в том, чтобы подобрать такие значения параметров $\mathbf{a_s}$, что функция $\mathbf{f(x, a_1, a_2, \ldots a_m)}$ наилучшим образом описывала бы исходную функцию. Как правило, $\mathbf{m<<n}$, поэтому добиться, чтобы функция $\mathbf{f(x, a_1, a_2, \ldots a_m)}$ давала точные результаты даже в узловых точках не удастся. Нужен критерий, который оценивает точность аппроксимации таблично заданной функции. Например, в методе наименьших квадратов в качестве такого критерия используется **среднеквадратическое отклонение**

$$\sqrt{\frac{(y_k - f(x_k, a_1, a_2, \ldots, a_m)^2}{m}} < \varepsilon .$$

Частным случаем задачи аппроксимации является задача **интерполяции** функции. В этом случае также имеем набор узловых точек $\mathbf{x_k}$ (где $\mathbf{k=1,2,\ldots,n}$) и значения функции $\mathbf{y_k}$ в этих точках. Однако, в соответствии с критерием интерполяции, требуется построить такую функцию **f(x),** которая в узловых точках $\mathbf{x_1, x_2,\ldots,x_n}$ принимала бы значения $\mathbf{y_1, y_2, \ldots,y_n}$, то есть $\mathbf{f(x_k) = y_k}$ для всех **k** от **1** до **n**.Чаще всего функцию **f(x)** ищут в виде полинома, степень которого **n-1**. Поэтому задача сводится к определению коэффициентов интерполяционного полинома на основании значений функции в базовых точках.

На практике для решения задачи интерполяции (вычисления значений функции в точках, несовпадающих с узлами интерполяции) используются **интерполяционные формулы Ньютона** и **формула Лагранжа** [1].

Для выполнения полиномиальной аппроксимации в MatLab используется функция **polyfit()**. Эта функция предназначена для выполнения ***аппроксимации методом наименьших квадратов***. Функция **polyfit(x,y,n)** возвращает вектор коэффициентов полинома степени **n**, который с наименьшей среднеквадратичной погрешностью аппроксимирует функцию, заданную таблично. Результатом является вектор строка длиной **n+1**, содержащий коэффициенты полинома в порядке уменьшения степеней. Аргументами функции являются список узловых точек, список значений интерполируемой функции в этих точках и степень интерполяционного полинома. Как правило, степень полинома много меньше количества узлов (**m<<n**). В качестве результата возвращается список коэффициентов аппроксимирующего полинома.

Ниже приведены пример линейной и кубической аппроксимации функции, заданной таблицей (рис. 2.2.1-1), и графики аппроксимирующих функций (рис. 2.2.1-2).

```
Command Window
>> X=[0 4 10 15 21 29 36 51 68];
>> Y=[60.7 61.4 60.2 59.8 58.7 60.0 60.8 59.8 59.7];
>> x=0:68;
>> [a]=polyfit(X,Y,1) %Вычисление коэффициентов линейного полинома
a =
   -0.0131   60.4623
>> ya1=a(1)*x+a(2);
>> [b]=polyfit(X,Y,3) %Вычисление коэффициентов  полинома 3-го пор
b =
   -0.0000    0.0048   -0.1513   61.1665
>> ya3=b(1)*x.^3+b(2)*x.^2+b(3)*x+b(4);
>> plot(X,Y,'*k',x,ya1,'-r',x,ya3,'-g')
>> grid on
```

Рис. 2.2.1-1. Пример линейной и кубической аппроксимации функции

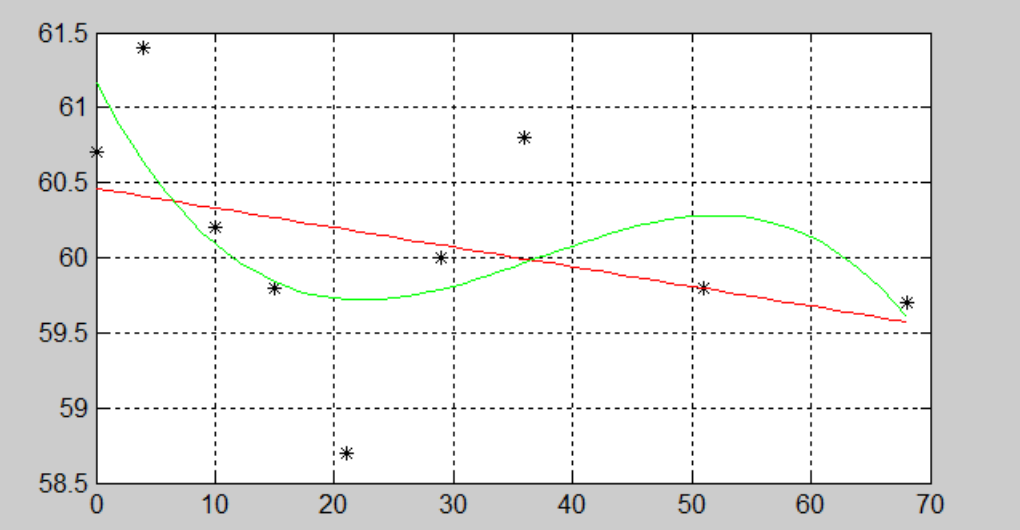


Рис. 2.2.1-2. Графики исходной и аппроксимирующих функций

Чтобы результатом выполнения функции Matlab**polyfit()** был именно ***интерполяционный полином,*** необходимо, чтобы степень этого полинома была на единицу меньше количества узловых точек.

Пример 2.2.1-1. Используя в качестве узлов интерполяции x=-5,-4…5, построить полином, интерполирующий функцию $y(x)=\frac{Sin(pi\cdot x/2)}{1+x^2}$.

Выполняя команды Matlab в окне **Command Window**, получим таблицу значений функции и проведем ее интерполяцию с использованием

функции Matlab **polyfit()** (рис. 2.2.1-3). Графики интерполяционных узлов и интерполяционного полинома приведены на рис. 2.2.1-4.Здесь для вычисления значений полинома в точках используется функция **polival(p,z)**, где вектор **z** покрывает интервал интерполяции и даже выходит за его пределы.

```
Command Window
>> x=-5:1:5;   %создание узлов интерполяции
>> y=sin(pi*x./2)./(1+x.^2);   %значения функции в узлах интерполяции
>> p=polyfit(x,y,length(x)-1);  %создание коэффициентов ИП полинома
>> p
p =
  Columns 1 through 8
   -0.0000    0.0001    0.0000   -0.0028   -0.0000    0.0516    0.0000   -0.3703
  Columns 9 through 11
   -0.0000    0.8215   -0.0000
>> z=-5.1:0.01:5.1; %узловые точки для построения графика
>> plot(x,y,'ks') %отображения точек графика в узлах ИП
>> grid on
>> plot(z,polyval(p,z),'r-')  %построение графика ИП полинома
>> title('Интерполировние функции')
```

Рис. 2.2.1-3. Интерполяция таблично заданной функции с использованием функции Matlab **polyfit()**

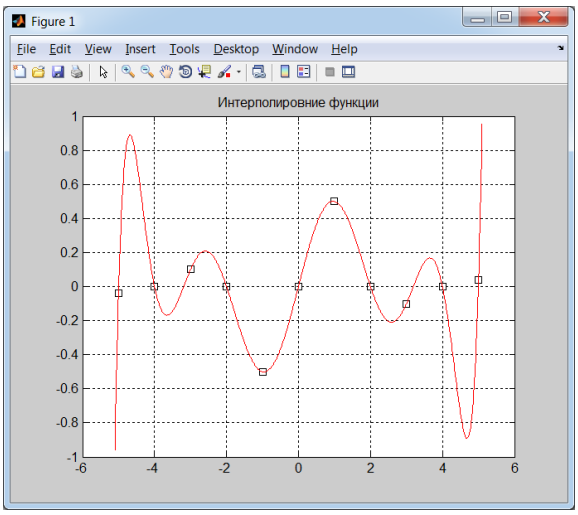


Рис. 2.2.1-4. Графики интерполируемой и интерполирующей функций

При большом количестве базовых точек интерполяция полиномом может оказаться малопродуктивной, поэтому нередко используют интерполяцию ***сплайнами***. Идея ***сплайн-интерполяции*** состоит в разбиении диапазона интерполирования на отрезки, в пределах которых используются разные функции одного вида (чаще всего алгебраические многочлены). Эта функция и ее несколько производных на всем диапазоне интерполяции

непрерывны. В результате имеем кусочно-гладкую интерполяционную зависимость.

Сплайн-интерполяцию можно реализовать с помощью функции Matlab**interp1()**. В качестве аргументов функции передают: набор узловых точек аргумента, значения функции в этих точках, список значений точек, для которых вычисляется значения интерполяционной зависимости и, наконец (в апострофах), тип базового полинома (табл. 2.2.1-1).

Таблица 2.2.1-1

Имя функции	Описание
nearest()	Интерполяция полиномами нулевой степени – график имеет ступенчатый вид
linear()	Интерполяция полиномами первой степени – базовые точки соединяются отрезками прямой
spline()	Интерполяция полиномами третьей степени

Рассмотрим пример использования функции Matlab **interp1().**

Пример 2.2.1-2. Используя значения интерполирующей функции, заданной таблично,

x	-1	0	1	2
y(x)	4	2	0	1

выполнить сплайн-интерполяцию с использованием полиномов нулевой, первой и третьей степени, и получить значения функции в точке x=0.58.

```
Command Window
>> x=[-1,0,1,2];
>> y=[4,2,6,10];
>> f1=interp1(x,y,0.58,'nearest')
f1 =
     6
>> f2=interp1(x,y,0.58,'linear')
f2 =
    4.3200
>> f3=interp1(x,y,0.58,'spline')
f3 =
    3.9741
```

Рис. 2.2.1-5. Вычисление функции в точке **x=0.58**

На рис. 2.2.1-6 и 2.2.1-7 приведены команды, необходимые для проведения интерполяции таблично заданной функции и построения графиков интерполируемой функции и интерполирующих ее интерполяционных многочленов различных степеней.

```
Command Window
>> x=[-1,0,1,2];
>> y=[4,2,6,10];
>> t=-1:0.2:2;
>> f1=interp1(x,y,t,'nearest');    %интерполяция 0-й степени
>> f2=interp1(x,y,t,'linear');     %интерполяция 1-й степени
>> f3=interp1(x,y,t,'splin');      %интерполяция 3-й степени
>> plot(x,y,'ks',t,f1,'r--',t,f2,'g-',t,f3,'m-.')
```

Рис. 2.2.2-6. Команды построения графиков интерполяционных функций

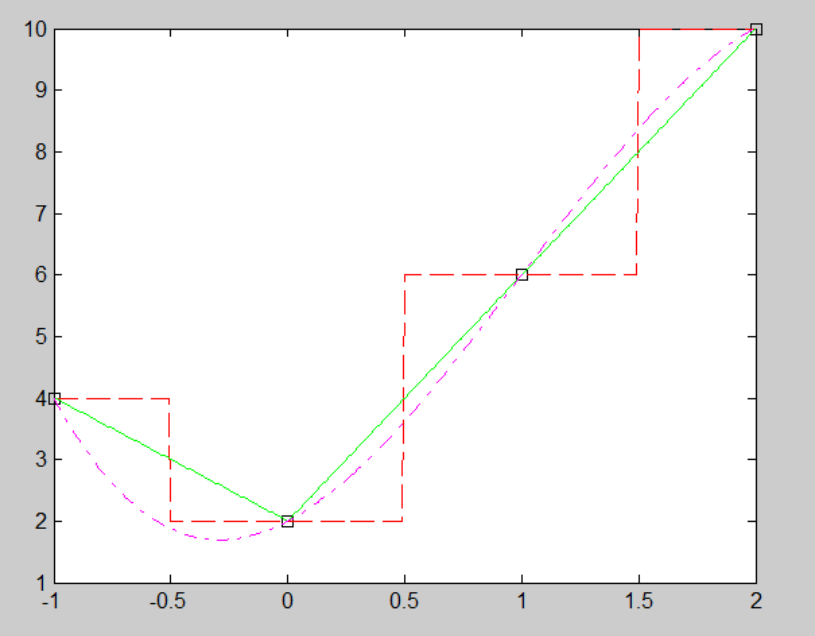


Рис. 2.2.2-7. Графики интерполирующих функций

2.2.2. Лабораторная работа по теме «Технология аппроксимации интерполяции функций»

1. Вопросы, подлежащие изучению

1) Задание векторов и матриц в пакете Matlab.
2) Технология аппроксимации функции, заданной таблично, с использованием функций **polyfit**().
3) Технология линейной, кубической и сплайн-интерполяции таблично заданной функции с использованием функций **interp1().**
4) Получение интерполяционных многочленов в явном виде.
5) Построение графиков аппроксимирующих и интерполирующих функций.

2. Общее задание

1) ***Изучите материал Темы 2.2. (п. 2.2.1)***.
2) ***Выберите индивидуальное задание****: номера узлов и номер аппроксимируемой функции из* ***табл. 2.2.2-1****; узлы аппроксимации и значения функции в узлах из* ***табл. 2.2.2-2****.*
3) ***Задайте*** *в виде векторов значения узлов и значения функции в выбранных узлах.*
4) ***Вычислите коэффициенты*** *аппроксимирующих функций для линейной, квадратичной и кубической аппроксимации с использованием функции* ***polyfit()*** *и* ***получите*** *три аппроксимирующие функции в явном виде.*
5) ***Получите*** *с использованием этих функций значение аппроксимирующей функции в произвольной точке, принадлежащей отрезку, но не совпадающей с узлами аппроксимации, и* ***сравните полученные*** *результаты.*
6) ***Постройте графики*** *табличной и трех аппроксимирующих функций в одном шаблоне, снабдив их легендой.*
7) ***Проведите*** *линейную и кубическую интерполяцию функции с использованием функции* ***interp1()****, заданной таблично. Получив значения интерполирующей функции в точке, не совпадающей с узлами интерполяции,* ***сравните*** *полученные результаты.*
8) ***Постройте*** *графики табличной и двух интерполирующих функций в одном шаблоне, снабдив их легендой.*
9) ***Представьте*** *результаты работы преподавателю,* ***ответьте*** *на поставленные вопросы.*
10) ***Выполните*** *команду* ***clear all***.
11) ***Оформите отчет*** *по выполненной работе****.***

3. Варианты заданий

Таблица 2.2.2-1

Вариант №	**Номера узлов x_i**	**Номер функции**
1	1 3 5 7 9 10 13	$f1(x)$
2	1 2 4 5 7 10 12	$f1(x)$
3	1 3 6 7 10 11 13	$f1(x)$
4	1 2 4 7 9 11 13	$f1(x)$
5	3 6 7 9 10 11 12	$f1(x)$
6	2 3 6 8 9 10 13	$f1(x)$
7	1 4 5 7 9 11 12	$f1(x)$
8	1 2 4 7 9 12 13	$f1(x)$
9	2 3 5 7 8 11 12	$f1(x)$
10	1 3 6 7 9 10 13	$f1(x)$
11	1 3 7 8 10 11 13	$f2(x)$
12	1 2 5 6 7 10 12	$f2(x)$
13	1 4 5 8 10 12 13	$f2(x)$
14	1 3 5 7 9 10 13	$f2(x)$
15	1 3 6 7 8 10 13	$f2(x)$
16	1 4 5 7 9 11 12	$f2(x)$
17	2 4 5 6 8 12 13	$f2(x)$
18	1 4 5 7 9 11 12	$f2(x)$
19	1 4 5 8 10 11 12	$f2(x)$
20	2 4 5 6 8 12 13	$f2(x)$
21	1 4 5 8 10 12 13	$f3(x)$
22	2 3 6 8 9 10 13	$f3(x)$
23	1 3 5 8 10 12 13	$f3(x)$
24	1 4 5 7 9 11 12	$f3(x)$
25	2 4 5 6 8 12 13	$f3(x)$
26	3 4 5 7 8 9 12	$f3(x)$
27	3 5 8 10 11 12 13	$f3(x)$
28	2 4 7 9 10 11 13	$f3(x)$
29	2 4 5 7 8 10 12	$f3(x)$
30	1 4 5 7 9 11 13	$f3(x)$

Таблица 2.2.2-2

i	x_i	$f1(x_i)$	$f2(x_i)$	$f3(x_i)$
1	-5	1.38	2.44	1.676
2	-4.5	1.221	2.359	2.025
3	-4	1.511	1.751	1.736
4	-3.5	1.501	2.13	1.203
5	-3	1	1.455	1.511
6	-2.5	0.728	1.482	1.362
7	-2	0.976	1.437	0.75
8	-1.5	1.065	0.803	0.976
9	-1	0.599	1.175	0.957
10	-0.5	0.192	0.49	0.272
11	0	0.3	0.375	0.3
12	0.5	0.319	$-6.51*10^{-3}$	0.165
13	1	-0.405	-1.965	-1.185

4. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в окне **Command Window**, снабженный необходимыми комментариями.

2.2.3. Контрольные вопросы по теме

1) Что такое аппроксимация функции и в каких случаях она используется?
2) В чем отличие аппроксимации от интерполяции?
3) Какой метод аппроксимации реализован в функции **polyfit()?**
4) Что служит результатом выполнения функции **polyfit()?**
5) Для чего предназначена функция **polyval()?**
6) Назначение и формат функции **interp1()?**
7) Каким параметром определяется тип интерполяции в функции **interp1()?**

Тема 2.3. Технология интегрирования в среде MatLab

2.3.1. Вычисление неопределенных и определенных интегралов
2.3.2. Лабораторная работа по теме «Технология интегрирования в среде MatLab»
2.3.3. Контрольные вопросы по теме

2.3.1. Вычисление неопределенных и определенных интегралов

При вычислении определенных интегралов первообразную функцию **F(x)** не всегда удается выразить аналитически, а кроме того иногда подынтегральная функция **f(x)** задана в виде таблицы (**x_i и y_i, где i = 1, 2, …, n**). Это приводит к необходимости использования численных методов интегрирования.

Существует ряд методов численного интегрирования. Во всех этих методах вычисление осуществляется по приближенным формулам, называемым **квадратурами**.

Суть получения формул численного интегрирования состоит в том, что на элементарных отрезках интегрирования подынтегральную функцию заменяют простейшим интерполяционным полиномом, который легко может быть проинтегрирован в аналитическом виде. Так, например, для получения формул **прямоугольников, трапеций** и **Симпсона** используют полиномы соответственно нулевой, первой и второй степени.

Формулы прямоугольников:

$$\int_a^b f(x)dx = \left(h\sum_{i=0}^{n-1} y_i \right),$$

где:

h – шаг интегрирования;

y_i – значение подынтегральной функции от аргумента **x_i, k=0, 1, …,n**;

n – число разбиений интервала интегрирования **a**, **b**.

Формула трапеций:

$$\int_a^b f(x)\,dx = \frac{h}{2}\left(y_0 + 2\cdot\sum_{i=1}^{n-1} y_i + y_n \right),$$

где:

h – шаг интегрирования;

y_0 – значение подынтегральной функции при **x = a**;

y_n – значение подынтегральной функции при **x = b**;

Формула Симпсона:

$$\int_a^b f(x)dx = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + ...2y_{n-2} + 4y_{n-1} + y_n).$$

Для символьного вычисления неопределенных и определенных интегралов используется функция Matlab **int()**, которая может иметь один из следующих форматов:

- **int(S)** – возвращает символьное значение неопределенного интеграла от символьного выражения или массива символьных выражений **S** по переменной, которая автоматически определяется функцией **findsym()**. Если **S** – скаляр или матрица, то вычисляется интеграл по переменной **'x'**.
- **int(S, v)** – возвращает неопределенный интеграл от **S** по символьной переменной **v**.
- **int(S, a, b)** – возвращает определенный интеграл от S с пределами интегрирования от **a** до **b,** причем пределы интегрирования могут быть как символьными, так и числовыми.
- **int(S, v, a, b)** – возвращает определенный интеграл от **S** по переменной **v** с пределами от **a** до **b.**

Ниже приведены примеры использования функции Matlab**int()** (рис. 2.3.1-1).

```
Command Window
>> syms x
>> int(sin(x)^3)    %вычисление символьного значения неопределенного интегр
ans =
(cos(x)*(cos(x)^2 - 3))/3
>> int([x^3 sin(x) exp(x/2)],x) %неопределенные интегралы от массива  выра
ans =
[ x^4/4, -cos(x), 2*exp(x/2)]
>> int(sin(x)^3,1,pi)    %возвращает определенный интеграл
ans =
-((cos(1) + 1)^2*(cos(1) - 2))/3
>> int(sin(x)^3,x,0,pi/2)
ans =
2/3
```

Рис. 2.3.1-1. Примеры вычисления интегралов

В системе MatLab функции вычисления интегралов используют численные методы **трапеции**, **Симпсона** и некоторые другие. Рассмотрим технологию интегрирования с использованием некоторых функций.

Для вычисления интеграла по **формуле трапеции** в MatLab используется функция **trapz(x,y)**. Эта функция возвращает значение

интеграла $\int_a^b y(x)\partial x$ от функции y(x), которая может быть представлена вектором или матрицей. Если **y(x)** – матрица, то функция возвращает вектор значений интеграла каждого столбца матрицы. Если вектор узлов не задан - **trapz(y)**, то в качестве ординат **x** используются индексы вектора **y**(**x=1:length(y)**), где функция определяет длину вектора **y**. Важно, что узлы по оси **x** могут быть как равноотстоящими, так и неравноотстоящими.

Рассмотрим несколько примеров вычисления значений определенных интегралов методом трапеций при различных способах задания узлов подынтегральной функции (рис. 2.3.1-2).

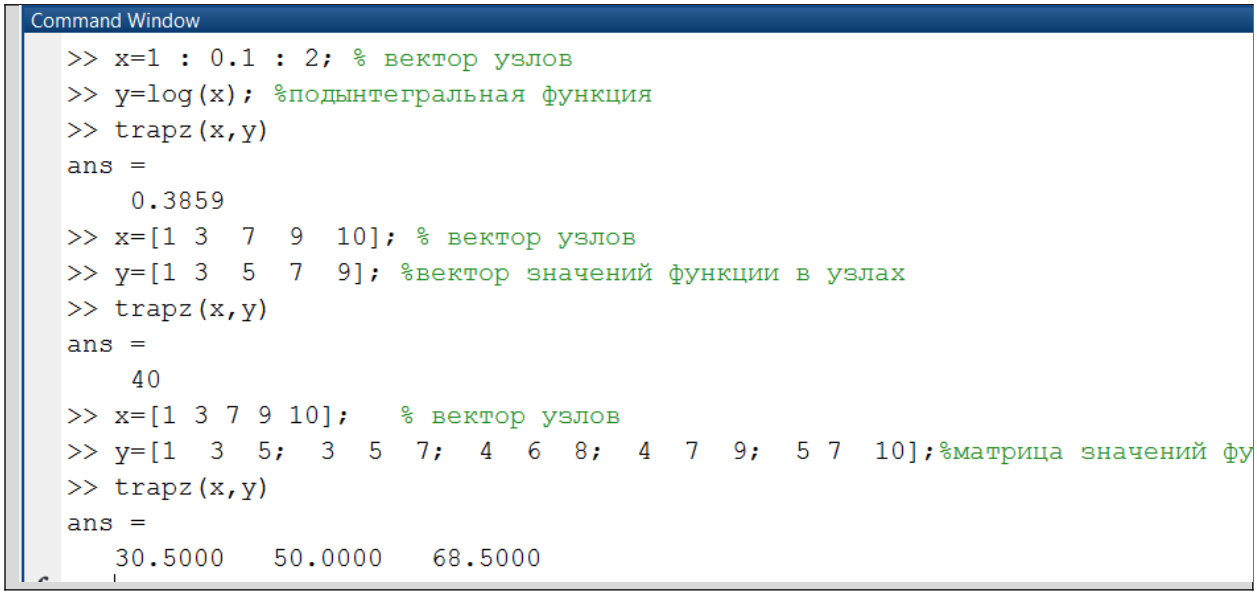


Рис. 2.3.1-2. Вычисление определенных интегралов с использованием ФункцииMatlab **trapz()**

Для вычисления интеграла по **формуле Симпсона** в MatLab применяется функция **quad()**. При обращении к этой функции шаг интегрирования не задается, а используется параметр – требуемая точность вычисления интеграла.

Минимальная форма обращения к функции – **q=quad('f',a,b)**, где **f** – имя функции, взятое в одинарные кавычки, второй и третий аргументы – пределы интегрирования.

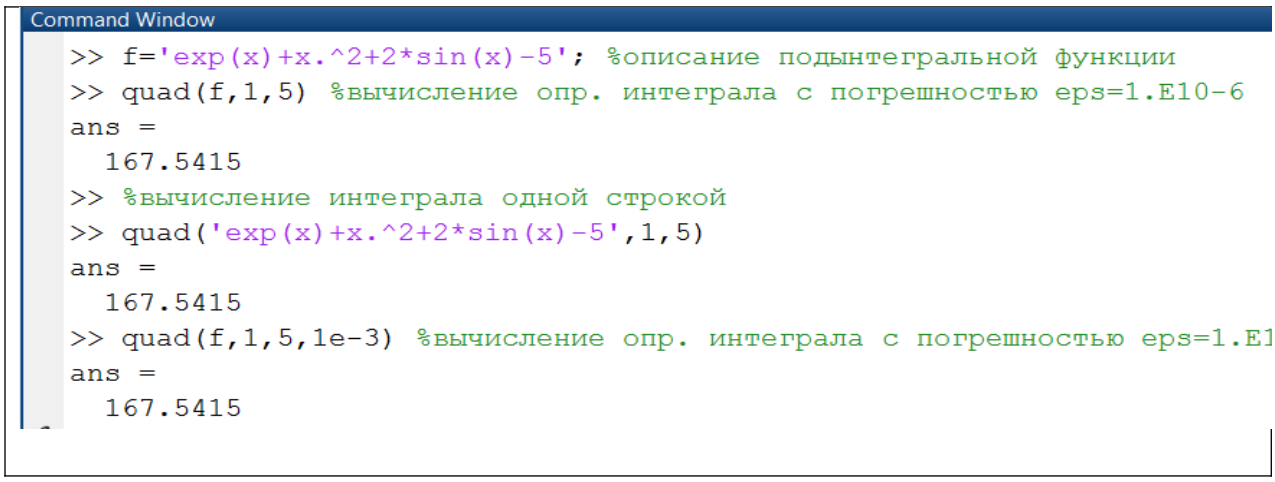


Рис. 2.3.1-3. Вычисление определенных интегралов
с использованием функции **quad()**

Допускается задание четвертого входного параметра **eps** – абсолютной погрешности: **q=quad('f', a, b, eps*)*. По умолчанию **eps =10**$^{-6}$.

Рассмотрим примеры вычисления определенного интеграла $\int_1^5 e^x + 2\,Sin(x) - 5\,\partial x$ с использованием функции Matlab **quad()** (рис. 2.3.1-3).

2.3.2. Лабораторная работа по теме «Технология интегрирования в среде Matlab»

1. Вопросы, подлежащие изучению

1) Получение символьного выражения неопределенного интеграла средствами пакета Matlab.
2) Вычисление значения определенного интеграла с использованием функций.
3) Вычисление значения определенного интеграла с использованием функций Matlab **int(), trapz(), quad().**
4) Формулы численного интегрирования: средних прямоугольников, трапеций и Симпсона.

2. Общее задание

1) ***Изучите*** *материал* ***Темы 2.3 (п. 2.3.1)****.*
2) ***Выберите*** *из* ***табл. 2.3.2-1*** *вариант индивидуального задания.*
3) ***Получить*** *символьное выражение неопределенного интеграла с использованием функций* ***int(S)*** *и* ***int(S, x)****, где* ***S*** *– символьное выражение, а* ***x*** *- переменная.*
4) ***Вычислите*** *значение определенного интеграла с использованием функции* ***int()*** *формата* ***int(S,a,b)****, где* ***a, b*** *– пределы интегрирования.*
5) ***Вычислите*** *определенный интеграл с использованием функции Matlab, реализующей формулу трапеций, -* ***trapz(x, y),*** *предварительно получив таблицу значений подынтегральной функции с шагом h и задав значения аргумента и функции в виде векторов.*
6) ***Вычислите*** *значение определенного интеграла с использованием функции Matlab, реализующей формулу Симпсона, –* ***quad('f',a,b),*** *где f – имя функции, взятое в одинарные кавычки.*
7) ***Представьте*** *результаты работы преподавателю,* ***ответьте*** *на поставленные вопросы.*
8) ***Выполните*** *команду* ***clear all.***
9) ***Оформите отчет*** *по выполненной работе****.***

3. Варианты индивидуальных заданий

Таблица 2.3.2-1

№	Интеграл 1	Интеграл 2
1	$\int \left(Cos(x)+e^{x}\right)dx$	$\int_{0}^{4}\left(8e^{-x}\,Sin(-2x)\right)dx$
2	$\int \frac{x}{1+x^{4}}\partial x$	$\int_{1}^{3}\left(Cos(x/5)(1+x)^{1/2}-x\right)\partial x$
3	$\int \frac{1}{x(1+x^{2})}\partial x$	$\int_{3.5}^{5}\left(5\ln(x)-x/2\right)\partial x$
4	$\int \frac{1}{\sqrt{(2+x)^{2}}}\partial x$	$\int_{0.5}^{1,5}\left(x^{2}+\ln(x)-2\right)\partial x$
5	$\int \left(-8\,Sin(x)+e^{-x}\right)\partial x$	$\int_{1.5}^{2.5}\left(1+Sin(4x)/\ln(x)\right)\partial x$
6	$\int \left(Cos(x^{2}+3x)\right)\partial x$	$\int_{2}^{3.5}\left(Sin(x^{2})+Cos^{2}(x)-10x\right)\partial x$
7	$\int \frac{x^{2}}{\sqrt{3+2x}}\partial x$	$\int_{0.5}^{3}\left(3Cos(x^{2})/\ln(x+3)\right)\partial x$
8	$\int \left(Sin(x^{-2})+x\right)\partial x$	$\int_{1}^{2.5}\left(x+Sin(x)/e^{x}\right)\partial x$
9	$\int \left(x^{2}+Cos^{2}(x+3)-x\right)\partial x$	$\int_{2}^{3.5}\left(Cos^{2}(x)-Sin(x)+x\right)\partial x$
10	$\int \frac{x}{1-x^{2}}\partial x$	$\int_{-1}^{0}\frac{x}{\sqrt{1-x}}\partial x$
11	$\int \frac{1}{x\sqrt{2+x^{2}}}\partial x$	$\int_{0}^{1.25}\frac{x^{2}}{\sqrt{1+x^{2}}}\partial x$
12	$\int \left(Sin(x)Cox^{2}(x)\right)\partial x$	$\int_{0}^{1}\frac{x^{2}}{\sqrt{1+x}}\partial x$
13	$\int \frac{1}{Sin(x)}\partial x$	$\int_{0}^{1.2}xCos(x)\partial x$
14	$\int \frac{1}{Cos(x)}\partial x$	$\int_{1}^{2.5}\left(x+Sin(x)/x^{2}\right)\partial x$
15	$\int \frac{1}{Sin^{2}(x)}\partial x$	$\int_{0.6}^{1.8}\frac{x}{\sqrt{0.8+x^{2}}}\partial x$
16	$\int \frac{x}{x^{2}(x-1)}\partial x$	$\int_{0.5}^{1.5}\frac{1}{\sqrt{x^{2}+2}}\partial x$
17	$\int \left(x+Cox^{2}(x)\right)\partial x$	$\int_{1}^{2.5}\left(Cos(x)/e^{x}\right)\partial x$
18	$\int \frac{x}{Cos(x)}\partial x$	$\int_{0.4}^{1.2}\frac{Cox(x^{2})}{x+1}\partial x$
19	$\int \frac{1}{Sin(x)Cos(x)}\partial x$	$\int_{0.4}^{3.2}x^{2}\ln(x)\partial x$

20	$\int (xCox^2(x)\partial x$	$\int_{0.2}^{0.8} \sqrt{x+1}\,Cos(x)\partial x$
21	$\int (2x+Cox^2(2x)\partial x$	$\int_{0.7}^{2.1} \frac{Sin(x)}{x+1}\partial x$
22	$\int Cos(x)+\frac{3}{Sin(x)}\partial x$	$\int_{1}^{2.5} (x+1)\,Cos(x^2)\partial x$
23	$\int \frac{Sin(x)}{Cos^4(x)}\partial x$	$\int_{0.4}^{1.2} \sqrt{x}\,Cos(x^2)\partial x$
24	$\int \frac{x^2}{e^x}dx$	$\int_{0.7}^{2.1} \frac{Sin(2x)}{x}\partial x$
25	$\int \frac{x^2}{\sqrt{1+2x}}dx$	$\int_{0.4}^{1.2} \frac{\cos(x)}{x+2}dx$
26	$\int (2x+1)\cdot e^{3x}dx$	$\int_{0}^{\pi} \frac{1}{3+2\cos(x)}dx$
27	$\int \frac{1}{1+\sin^2(x)}dx$	$\int_{-1}^{1} \frac{1}{(1+x^2)^2}dx$
28	$\int \frac{1}{\sqrt{4x^2+2x+1}}dx$	$\int_{1}^{5} \frac{\sqrt{x-1}}{x}dx$
29	$\int \frac{\sqrt{4-x^2}}{x^2}dx$	$\int_{1}^{3} \frac{1}{x\sqrt{x^2+1}}dx$
30	$\int \cos^2(x)\sin^3(x)dx$	$\int_{1}^{2} \frac{1-e^{2x}}{x\cdot e^{2x}}dx$

4. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в окне **Command Window**, снабженный необходимыми комментариями**.**

2.3.3. Контрольные вопросы по теме

1) Какой функцией в Matlab определяется символьное значение определенного интеграла?
2) Назначение функции **trap(x,y).**
3) Что возвращает функция **trap(x,y),** если **y(x)** – матрица?
4) Можно ли использовать функцию **trap(x,y),** если узлы по оси **x** - не равноотстоящие?
5) Какая функция Matlab позволяет вычислить определенный интеграл с заданной точностью?
6) Способы задания подынтегральной функции при вычислении определенного интеграла с использованием функции **quad().**
7) Какова точность вычисления определенного интеграла по умолчанию?

Тема 2.4. Технология решения обыкновенных дифференциальных уравнений

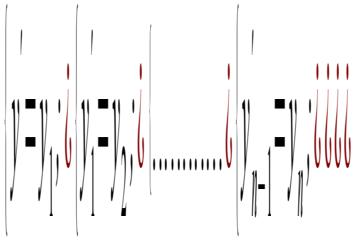
2.4.1. Численное решение обыкновенных дифференциальных уравнений
2.4.2. Лабораторная работа по теме
2.4.3. Контрольные вопросы по теме

2.4.1. Численное решение обыкновенных дифференциальных уравнений

Представим обыкновенное дифференциальное уравнение (ОДУ) первого порядка в виде, разрешенном относительно производной $y' = f(x, y)$, и пусть $y(x_0) = y$ – начальные условия его решения.

Тогда решением ОДУ является функция $y = \phi(x)$, которая, будучи подставленной в исходное уравнение, обратит его в тождество, и одновременно будут выполняться начальные условия. Эта задача в математике называется ***задачей Коши***.

Задача Коши при решении ОДУ n-го порядка $y^{(n)} = f(x, y, y', \ldots y^{(n-1)})$ формулируется аналогично, при этом начальными условиями должны быть: $y(x_0) = y_0, y'(x_0) = y_0', \ldots \quad y^{(n-1)}(x_0) = y_0^{(n-1)}$. При решении ОДУ n-го порядка уравнение путем выполнения ряда обозначений $(y' = y_1, y_1' = y_2, \ldots \quad y_{n-2}' = y_{n-1})$ представляется в виде системы дифференциальных уравнений:



Результатом решения **ОДУ** численными методами является ***таблица значений*** **y = φ(x)** на некотором множестве значений аргументов. Поэтому при постановке задачи численного решения **ОДУ** наряду с начальными условиями $\mathbf{x_0}$, $\mathbf{y_0}$ необходимо задать область решения – отрезок **[a;b]** и шаг изменения аргумента **h** (шаг интегрирования).

Для получения численного решения ОДУ используются **методы Рунге-Кутты** [1]. Методы различаются порядком. Чем выше порядок метода, тем точнее решение, полученное при равном шаге интегрирования.

В Matlab имеется несколько функций для решения задачи Коши. Рассмотрим две функции, используемые для решения обыкновенных

дифференциальных уравнений (ОДУ): **ode23()** – использует метод Рунге-Кутты второго и третьего порядка; **ode45()** – использует метод Рунге-Кутты четвёртого и пятого порядка точности с автоматическим выбором шага.

В MatLab обращением к функциям, предназначенным для решения ОДУ, является:

[x, y] = ode23('fun',t0,tf,x0)
[x, y] = ode45('fun',t0,tf,x0)

Входными параметрами функций являются:

- **'fun'** – имя функции (в виде строчной переменной) или ссылка на функцию, задающую правую часть дифференциального уравнения (уравнение должно быть записано в нормальной форме y' = **fun(x,y));**
- **t0** – начальное значение аргумента;
- **tf** – конечное значение аргумента**;**
- **x0** – вектор начальных условий.

Выходными параметрами функций являются:

- **x** – вектор, содержащий отсчёты аргумента в точках решения;
- **y** – вектор, содержащий вычисленные значения результата решения ОДУ в точках, соответствующих отсчетам независимой переменной **x**.

Требования к точности и другие параметры численного решения задаются в Matlab по умолчанию. Изменить эти настройки позволяет дополнительный аргумент **OPTIONS**.

```
Command Window
>> f=@(x,y)(y.^2-2./x.^2); %правая часть ОДУ
>> [x,y]=ode23(f,1:1:5,1) %решение ОДУ методом Рунке-Кутты 2-3 порядка
x =
     1
     2
     3
     4
     5
y =
    1.0000
    0.4986
    0.3297
    0.2436
    0.1900
```

Рис. 2.4.1-1. Решение ОДУ методом Рунге-Кутты второго порядка

Рассмотрим примеры использования функций **ode23()** и **ode45()** для решения ОДУ вида: $y' = y^2 - \frac{2}{x^2}$ на отрезке **[1;10]** с шагом интегрирования 1 при начальных условиях y(1)=1. Решение ОДУ с использованием функции **ode23()** приведено на рис. 2.4.1-1.

На рис. 2.4.1-2 приведено решение того же ОДУ, но с использованием функции **ode45()**. Вывод таблицы решения дополнен графиком функции y(x) (рис. 2.4.1-3).

```
Command Window
>> f=@(x,y)(y.^2-2./x.^2); %правая часть ОДУ
>> [x,y]=ode45(f,1:1:5,1) %решение ОДУ методом Рунке-Кутты 4-5 по
x =
     1
     2
     3
     4
     5
y =
    1.0000
    0.4999
    0.3332
    0.2498
    0.1997
>> [x,y]=ode45(f,1:0.1:5,1);
>> plot(x,y,'k-')
>> grid on
```

Рис. 2.4.1-2. Решение ОДУ методом Рунге-Кутты четвертого порядка

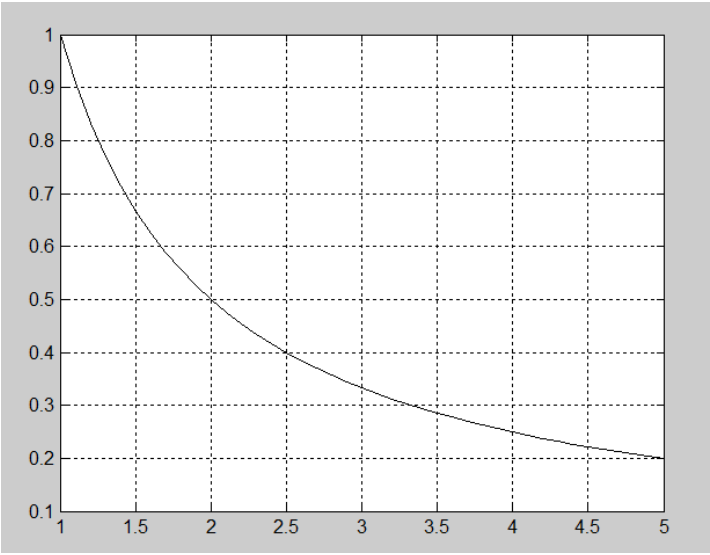


Рис. 2.4.1-3. Графическое решение ОДУ

При решении ОДУ высокого порядка вводят обозначения (новые функции) для всех производных, кроме самой старшей. В этих обозначениях самая старшая производная становится записанной, как первая. Исходное уравнение дополняется тождествами, определяющими правила введения новых функций. Проиллюстрируем описанный подход при решении **ОДУ второго порядка**.

Требуется решить $y''(x)+2y'(x)+2y(x)=0$ с начальными условиями: и $y'(0)=1$. Сведем заданное ОДУ к равноценной системе ОДУ первого порядка. Для этого переобозначим функцию $y(x)$, как $y_1(x)$, и введем функцию $y_2(x)=y_1'(x)$. Запишем исходное уравнение в новых обозначениях: $y_2'=-2(y_1(x)+y_2(x))$(здесь учтено, что $y_2''(x)=y_2'(x)$). Дополним это уравнение тождеством $y_1'(x)=y_2(x)$ и получим нужную систему ОДУ. Эту систему следует дополнить начальными условиями: $y_1(0)=2, y_2(0)=1$.

Решение системы ОДУ в Matlab начинается с определения векторной функции (рис. 2.4.1-4).

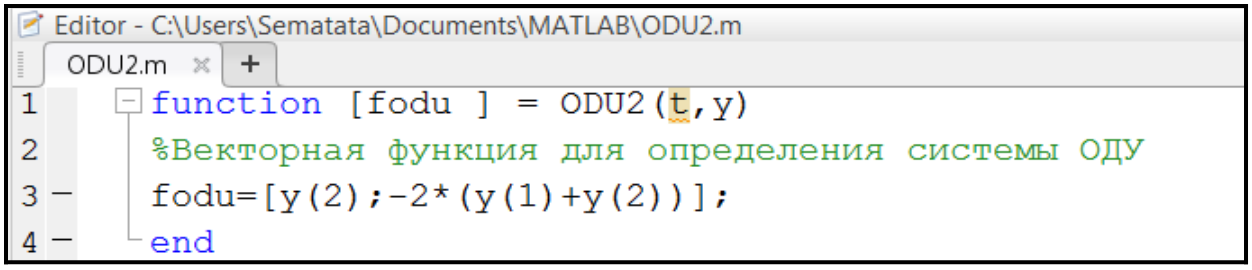


Рис. 2.4.1-4. Определение векторной функции

Для решения дифференциального уравнения (системы уравнений) используем команды, где решение системы ОДУ записывается в переменную **y**. При этом элемент **y(1)** содержит значение функции **y(x),** а **y(2)** значение производной **y'(x)**. Команда **plot(x, y)** строит график функции решения ОДУ и ее производной (рис. 2.4.1-6).

```
Command Window
>> [x,y]=ode45('ODU2',0:0.1:1,[2,1]);
>> y
y =
    2.0000    1.0000
    2.0716    0.4487
    2.0928   -0.0109
    2.0722   -0.3869
    2.0179   -0.6878
    1.9369   -0.9217
    1.8356   -1.0965
    1.7193   -1.2197
    1.5931   -1.2986
    1.4609   -1.3397
    1.3262   -1.3490
>> plot(x,y)
>> grid on
>> title('Решение ОДУ второго порядка')
>> legend('Функция y(x)','Производная y''(x)')
```

Рис. 2.4.1-5. Решение ОДУ второго порядка

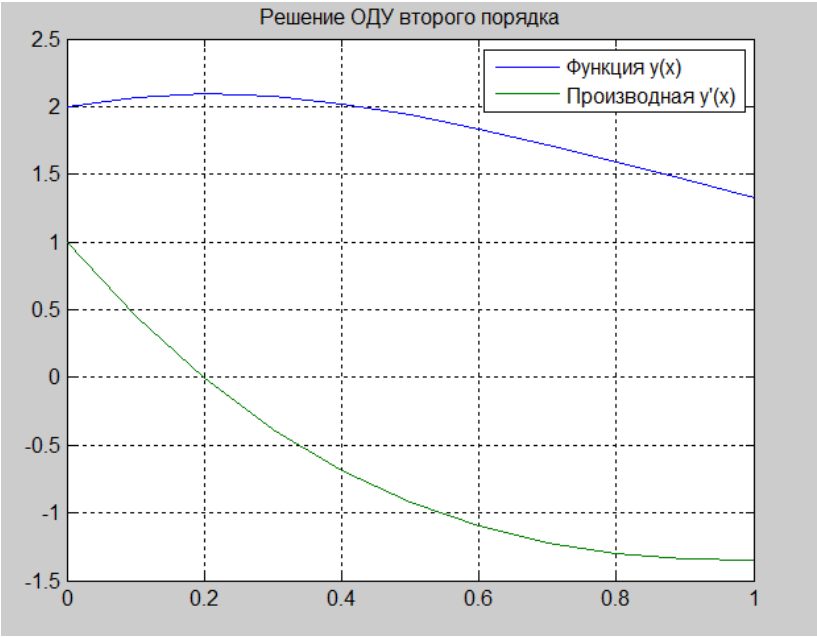


Рис. 2.4.1-6. Графики функций решения ОДУ и производной

2.4.2. Лабораторная работа по теме «Технология решения обыкновенных дифференциальных уравнений средствами MatLab»

1. Вопросы, подлежащие изучению

1) Нахождение символьного выражения производной от функции с использованием функции **diff()**.
2) Вычисление числовых значений производных в точках.
3) Функции **Matlab**, предназначенные для решения ОДУ (**ode23(), ode45()**).
4) Вывод результатов решения ОДУ в виде таблицы.
5) Получение графического решения ОДУ.

2. Общее задание

1) ***Изучите*** *материал* ***Темы 2.4 (п. 2.4.1)****.*
2) ***Выберите*** *индивидуальный вариант задания из* ***табл. 2.4.2-1****.*
3) ***Найдите*** *символьное выражение производной от функции f(x) и вычислите значение производной от функции f(x) в произвольной точке* ***c****.*
4) ***Найдите*** *решение ОДУ на отрезке [a;b] с шагом h с использованием функций* ***ode23()*** *и* ***ode45().***
5) ***Создайте*** *матрицу решений, записав в первый столбец аргумент, во второй и третий - решение, полученное с использованием функций* ***ode23()*** *и* ***ode45(),*** *а в четвертый столбец – погрешность метода (модуль разности решений ОДУ с использованием функций* ***ode23()*** *и* ***ode45()).***
6) ***Выведите*** *полученную таблицу по столбцам.*
7) ***Постройте*** *графики полученных решений ОДУ в одном шаблоне.*
8) ***Сохраните*** *текст рабочего окна на внешнем носителе.*
9) ***Представьте*** *результаты работы преподавателю,* ***ответьте*** *на поставленные вопросы.*
10) ***Выполните*** *команду* ***clear all.***
11) ***Оформите отчет*** *по выполненной работе****.***

2. Варианты индивидуальных заданий

Таблица 2.4.2-1

№	f(x)	ОДУ	Начальные условия	b	h
1	$8e^{-x}Sin(-2x)$	$y' = x + 2y$	$y(0) = 1$	0,5	0,5
2	$Sin(x+1)e^{2/x}$	$y' = 2\sqrt{xy}$	$y(1) = 1$	1.2	0.2
3	$3x - Cos(x) - 1$	$y' = x + e^{y}$	$y(0) = 0$	0.4	0.2
4	$tgx - \frac{7}{2x+6}$	$y' = \sqrt{y}\,x$	$y(0) = 1$	1	0,5
5	$(x-3)Cos(x) - 1$	$y' = \sqrt{y} + 2x$	$y(0) = 1$	0.4	0,2
6	$Sin(x+1) - 0,5x$	$y' = x^2 - 2y$	$y(0) = 0$	1	0.25
7	$x^2 Cos(2x) + 1$	$y' = e^{y} - 2x$	$y(0) = 1$	0.2	0,2
8	$(x-1)^2 \cdot \ln(x+11) - 1$	$y' = ye^{x/2}$	$y(0) = 1$	0.5	0,25
9	$Cos(x+0,5) - x^2$	$y' = e^{x} + x$	$y(0) = 1$	0,25	0.25
10	$\frac{x}{\sin(x)} + 2x^2$	$y' = y^2 - x$	$y(0) = 1$	1	0,5
11	$\frac{e^{x}}{2} - (x-1)^2$	$y' = e^{y} - 2x$	$y(0) = 1$	0,3	0,15
12	$e^{-6x} + 3x^2 - 18$	$y' = y + x^2 y$	$y(0) = 1$	0,5	0,25
13	$x - \ln(7x-4)$	$y' = 1.5x^2/2(y+1)$	$y(0) = 0$	0,5	0.5
14	$Sin(Sin(x)) - 0,5x$	$y' = (y-1)/x$	$y(0) = 1$	1	0,5
15	$x^2 Sin(\ln(x))$	$y' = 2xy - 3$	$y(1) = 1$	2	0,5
16	$4e^{-1/x} + x^3 - Cos(x)$	$y' = (x+y)^2$	$y(1) = 1$	2	0,5
17	$e^{-2x} + \frac{3}{x} + x^3$	$y' = xe^{-y}$	$y(1) = 0$	2	0,5
18	$2^{x} + \ln(2x) - 5,6x^4$	$y' = (x-y)^2$	$y(1) = 1$	2	0,5
19	$x^2 + Sin(x/2) + 1$	$y' = \cos(y) + x^2$	$y(1) = 1$	2	0,5
20	$2e^{-2x} Sin(2x)$	$y' = 2xy + x^2$	$y(1) = 1$	2	0,25
21	$x - \ln(7x-4)$	$xy' = x^2 - y$	$y(1) = 1$	2	0.25
22	$8e^{-x} Sin(-2x)$	$y' = y^2 - x$	$y(0) = 1$	1	0,5
23	$Sin(x+1)e^{2/x}$	$y' = x^2 - y$	$y(1) = 1$	1,5	0,5
24	$3x - Cos(x) - 1$	$y' = xy^2 + 1$	$y(1) = 1$	1.4	0.2
25	$tgx - 7/(2x+6)$	$xy' + y = x$	$y(1) = 1$	2	0,5
26	$Sin(x+1)e^{2/x}$	$xy' + y = x^2$	$y(1) = 1$	2	0,5
27	$x - \ln(7x-4)$	$xy' - y = x^2$	$y(1) = 1$	2	0,5
28	$Cos(x+0,5) - x^2$	$y' = y/(1+x)$	$y(0) = 1$	1,4	0,2
29	$(x-3)Cos(x) - 1$	$y' - x^2 y = x$	$y(1) = 1$	1,4	0,2

30	$Sin(x+1)e^{2/x}$	$x\,y' - y = x$	$y(1) = 1$	2	0.5

3. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в окне Command Window, снабженный необходимыми комментариями.

2.4.3. Контрольные вопросы по теме

1) Какие начальные условия должны быть заданы в соответствии с задачей Коши при решении ОДУ средствами системы Matlab?
2) Какие численные методы реализованы в функциях Matlab **ode23()** и **ode45()?**
3) Что является входными параметрами функций **ode23()** и **ode45()**?
4) В какой форме должны быть записаны функции ОДУ при использовании **ode23()** и **ode45()?**
5) Можно ли задать при решении ОДУ средствами Matlab требуемую точность вычислений?
6) Что представляет собой решение ОДУ второго порядка при использовании функций **ode23()** и **ode45()?**
7) Почему при решении ОДУ старших порядков необходимо использовать векторную функцию?

8)

Тема 2.5. Технология решения задач одномерной оптимизации

2.5.1. Решение задач одномерной оптимизации функций
2.5.2. Лабораторная работа по теме «Технология решения задач одномерной оптимизации»
2.5.3. Контрольные вопросы по теме

2.5.1. Решение задач одномерной оптимизации

При решении задачи поиска экстремума (максимума или минимума) функции **y=f(x)** одной переменной выделяют задачи поиска ***локального*** и ***глобального*** экстремума. При этом задача нахождения максимума целевой функции сводится к задаче нахождения минимума путем замены функции **f(x)** на **-f(x)**, поэтому в дальнейшем будем говорить только о поиске минимума функции, то есть такого $x^* \in [a, b]$, при котором $f(x^*) = \min f(x)$.

Интервал, на котором локализован единственный минимум, называется **отрезком неопределенности.**

Необходимым условием существования экстремума дифференцируемой функции **f(x)** является выполнение равенства $f'(x) = 0$. Точка **x**, удовлетворяющая данному условию, называется **точкой стационарности**. **Достаточным** условием существования минимума в точке стационарности является выполнение неравенства $f''(x)>0$, а максимума - $f''(x)<0$.

Задача одномерной оптимизации имеет единственное решение в том случае, если функция **f(x)** на отрезке **[a;b]** имеет только один экстремум. Тогда говорят, что функция унимодальна на отрезке **[a;b]**.

Достаточными условиями унимодальной функции на отрезке **[a;b]** являются:

- для дифференцируемой функции **f(x)** ее производная $f'(x)$ - неубывающая;
- для дважды дифференцируемой функции **f(x)** выполняется неравенство $f''(x)\geq 0$.

Для решения задачи одномерной оптимизации с заданной степенью точности используются методы: **дихотомии**, **золотого сечения**, **средней точки** и многие другие [1]. При этом суть методов одномерного поиска заключается в том, что на каждой итерации интервал неопределенности уменьшается и стягивается к точке минимума. Уменьшение отрезка происходит до тех пор, пока на некоторой **n-**й итерации отрезок неопределенности $[b_n;a_n]$ не станет соизмеримым с заданной погрешностью ε, то есть будет выполняться условие $|b_n-a_n| <\varepsilon$. Тогда за точку минимума

можно принять любую точку, принадлежащую этому отрезку, в частности, его середину.

В MatLab поиск локального минимума осуществляет функция **fminbnd()**, имеющая следующий формат:

[x, y] = fminbnd(name, a, b), где:

name – имя функции, вычисляющей значение f(x);

a, b – границы интервала, на котором осуществляется поиск минимума;

x, y – координаты точки минимума на заданном интервале.

Чтобы с использованием функции Matlab **fminbnd()** вычислить локальный ***максимум,*** необходимо взять целевую функцию с противоположным знаком.

Пример 2.5.1-1. Найти локальный минимум функции $f(x)=x^4-0.5x^3-28x^2+140$.

Решение задачи начнем с построения графика (рис. 2.5.1-1 и 2.5.1-2). Определим отрезок, содержащий точку минимума, исходя из вида графика функции. Если это возможно, то на этом отрезке следует провести полное исследование функции на унимодальность: от функции $f(x)$ получить и первую, и вторую производные и показать, что $f'(x)$ должна быть неубывающей, а $f''(x)>0$.

```
Command Window
>> f=@(x)(x.^4-0.5*x.^3-28*x.^2+140);
>> %построение графика
>> x=-5:0.1:6;
>> y=f(x);
>> plot(x,y,'r-')
>> grid on
```

Рис. 2.5.1-1. Построение графика функции f(x)

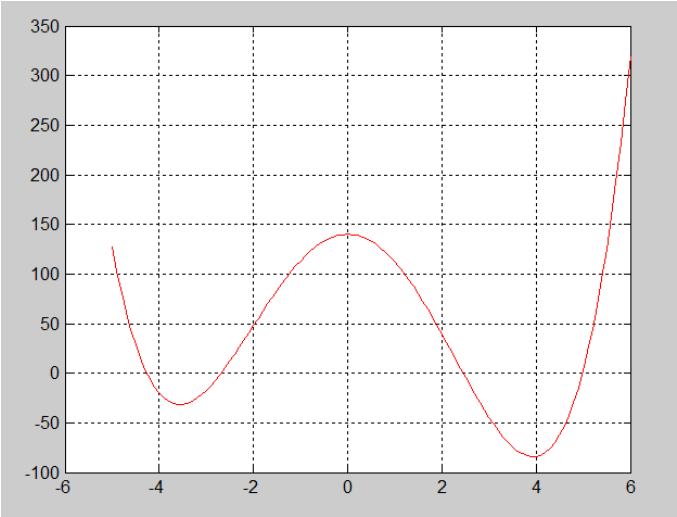


Рис. 2.5.9-2. График функции f(x)

Из графика следует, что на отрезках **[-4;-3]** и **[3;5]** имеются локальные минимумы. Исследуем функцию и поведение производных, например, на отрезке **[3;5**].

Для вычисления производных используем функцию Matlab **diff()**, имеющую следующий формат: **diff(f, x, n)**. В качестве параметров этой функции используется:

- **F** – дифференцируемая функция;
- **x** – аргумент функции (переменная дифференцирования);
- **n** – порядок производной.

Получение значений функции и производных на выбранном отрезке приведено на рис. 2.5.1-3.

```
Command Window
>> %вычисление первой и второй производных на [5;6]
>> syms z
>> ff=z^4-0.5*z^3-28*z^2+140;
>> diff(ff,z)
ans =
4*z^3 - (3*z^2)/2 - 56*z
>> diff(ff,z,2)
ans =
12*z^2 - 3*z - 56
>> %Таблица значений функции и прозводных
>> x=3:0.4:5;
>> t=[x;x.^4-0.5*x.^3-28*x.^2+140;4*x.^3-(3*x.^2)/2-56.*x;12*x.^2 - 3*x
t =
    3.0000    3.4000    3.8000    4.2000    4.6000    5.0000
  -44.5000  -69.6984  -83.2424  -79.7944  -53.4024    2.5000
  -73.5000  -50.5240  -14.9720   34.6920  100.0040  182.5000
   43.0000   72.5200  105.8800  143.0800  184.1200  229.0000
```

Рис. 2.5.1-3. Исследование функции f(x) на отрезке [3;5]

Из полученной таблицы значений функции и производных видно, что на отрезке [3;5] существует единственный минимум. Найдем координаты этого минимума с использованием функции Matlab **fminbnd()** (рис. 2.5.1-4).

Рис. 2.5.1-4. Нахождение координат точки минимума функции f(x)

2.5.2. Лабораторная работа по теме «Технология решения задач одномерной оптимизации»

1. Вопросы, подлежащие изучению

1) Условие унимодальности функции на отрезке.
2) Получение таблиц значений функции и ее производных с использованием средств пакета Matlab.
3) Технология использования встроенной функции пакета Matlab – **fminbnd().**

2. Общее задание

1) ***Изучите*** *материал* ***Темы 2.5 (п. 2.5.1)****.*
2) ***Выберите*** *индивидуальное задание из* ***табл. 2.5.2-1****.*
3) ***Постройте*** *график функции* ***f(x),*** *выберите отрезок, содержащий минимум.*
4) ***Проверьте*** *на выбранном отрезке условие унимодальности функции, получив таблицу значений первой или второй производной.*
5) ***Найдите*** *координаты точки минимума* ***f(x)*** *с использованием встроенной функции Matlab* ***fminbnd().***
6) ***Сохраните*** *текст рабочего окна на внешнем носителе.*
7) ***Представьте*** *результаты работы преподавателю,* ***ответьте*** *на поставленные вопросы.*
8) ***Выполните*** *команду* ***clear all.***
9) ***Оформите отчет*** *по выполненной работе.*

3. Варианты индивидуальных заданий

Таблица 2.5.2-1

№	f(x)	№	f(x)
1	$x^2-6\ln(x+1)+2,8$	16	$2x^2-\sin x-1,5$
2	$x^2-4\sin x+0,8$	17	$2\cos(x-x^3)-x$
3	$x^2-3\sin x-2,4$	18	$x^2-3,4\ln(1+x)-1$
4	$x^2+\sin x-0,5\sqrt[3]{(1+2x)}$	19	$e^x-(x+1)^2-2$
5	$3x^2-2\cos(x+1)-4,8$	20	$x^2-\sin x$

6	$x^2-1,5\cos(x+3)-1,8$	21	$e^x-4x-9,5$
7	$\ln(x+2)+\cos(2+x)-0,35$	22	$x^2-3\sin x-2x$
8	$x^3-4,8\sin(1+2x)-1$	23	$3-2\sin x-\sqrt{x^2+1}$
9	$x^2-2\sin(x+1)-2$	24	$-2+3\sin x-\sqrt{x^2+2}$
10	$e^{x-3}-\ln(x+3)+1$	25	$e^{-0,4x}+3x^2-5x-3$
11	$-\sin x-\sqrt{x}+1$	26	$x^2+\cos(x-4)+\sqrt{x+1}-3$
12	$\cos(x-0,2x^2)-x+6$	27	$x^2-\ln x-3$
13	e^x-4x	28	$-x+e^{0,5x}$
14	$0,1x^2+x\ln x-x+0,5$	29	$-\sin x-\sqrt{1+x}+2$
15	$2e^{-x}+e^x-4$	30	$4x^3+2x^4+9,5e^{-2x}-35$

1. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в окне Command Window.

2.5.3. Контрольные вопросы по теме

1) Понятия локального и глобального минимума функции.
2) Можно ли средствами Matlab вычислить глобальный минимум заданной функции?
3) Какие функции необходимо использовать перед поиском локального минимума?
4) Какая функция Matlab позволяет получить символьные выражения для производных от функции?
5) Назначение функции **diff(f, x, n)** и ее параметров.
6) Функция **fminbnd(name, a, b)** и назначение ее параметров.
7) Можно ли с использованием функции вычислить локальный максимум?

Тема 2.6. Технология решения задач многомерной оптимизации

2.6.1. Решение задач многомерной оптимизации функций
2.6.2. Лабораторная работа по теме «Технология решения задач многомерной оптимизации»
2.6.3. Контрольные вопросы по теме

2.6.1. Решение задач многомерной оптимизации

Задача, требующая нахождения оптимального значения функции m переменных $\mathbf{Q(X) = Q(x_1, x_2, \ldots, x_m)}$, называется задачей **многомерной оптимизации**. Так же, как и для случая одномерной оптимизации, задача нахождения максимума функции сводится к задаче нахождения минимума путем замены целевой функции **Q** на **-Q**.

В постановке задачи безусловной оптимизации для $\mathbf{Q(X)=Q(x_1,x_2,\ldots,x_m)}$ требуется найти хотя бы одну точку минимума **X*** и вычислить $\mathbf{Q^*=f(X^*)}$. Точка $\mathbf{X^* \in R^m}$ называется точкой глобального минимума функции **Q** на множестве **X**, если для всех $\mathbf{X \in R^m}$ выполняется неравенство $\mathbf{Q(X^*) \le Q(X).}$ В этом случае значение **Q(X*)** называется минимальным значением функции **Q** на $\mathbf{R^m}$. Точка $\mathbf{X^* \in R^m}$ называется точкой локального минимума функции **Q**, если существует такая **δ** - окрестность $\mathbf{U_\delta}$ этой точки (**δ>0**), что для всех $\mathbf{X \in X_\delta = X \cap U_\delta}$ выполняется неравенство **Q(X*) ≤ Q(X)**.

Для всякой непрерывно дифференцируемой функции Q достаточным условием того, что функция имеет точку минимума, является положительная определенность матрицы вторых частных производных (**матрицы Гессе**):

$$G(\bar{x}) = f''(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_m} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_m} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_m \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_m \partial x_m} \end{bmatrix}$$

Известно, что для того чтобы матрица была положительно определена, необходимо, чтобы все угловые миноры были положительны. Так, для функции двух переменных Q(x, y) матрица Гессе имеет вид:

$$G(x,y) = \begin{vmatrix} \frac{\partial^2 Q}{\partial x^2} & \frac{\partial^2 Q}{\partial x \partial y} \\ \frac{\partial^2 Q}{\partial y \partial x} & \frac{\partial^2 Q}{\partial y^2} \end{vmatrix},$$

а достаточным условием существования минимума является выполнение неравенств:

$$\Delta_1 = \frac{\partial^2 Q}{\partial x^2} > 0,$$

$$\Delta_2 = \frac{\partial^2 Q}{\partial x^2} \cdot \frac{\partial^2 Q}{\partial y^2} - \left(\frac{\partial^2 Q}{\partial x \partial y} \right)^2 > 0.$$

Аналитический метод поиска минимума применяется только для ограниченного круга задач. В основном это связано с необходимостью решения системы нелинейных уравнений, которая, как правило, решается численными методами. Гораздо проще решать задачу многомерной оптимизации численными методами, например, такими как **метод градиентного спуска с дроблением шага** и **методы наискорейшего спуска с аналитическим или численным выбором шага** [1].

Вычисление экстремума функции нескольких переменных

$$z = f(x_1, x_2, \ldots x_n)$$

в MatLab осуществляет функция:

[X, Z] = fminsearch(name, x0),

где:

- **name** – имя функции, зависящее от n переменных;
- **x0** – вектор из n элементов, содержащий координаты точки начального приближения;
- **X – вектор из n элементов, содержащий координаты точки, в которой достигается минимум;**
- **Z** – значение функции в точке с координатами **X**.

Рассмотрим работу функции **fminsearch()** на примере определения минимума двумерной функции $f(x1, x2) = x_1^2 + x_2^2 - x_2 - 2x_1 + 2$.

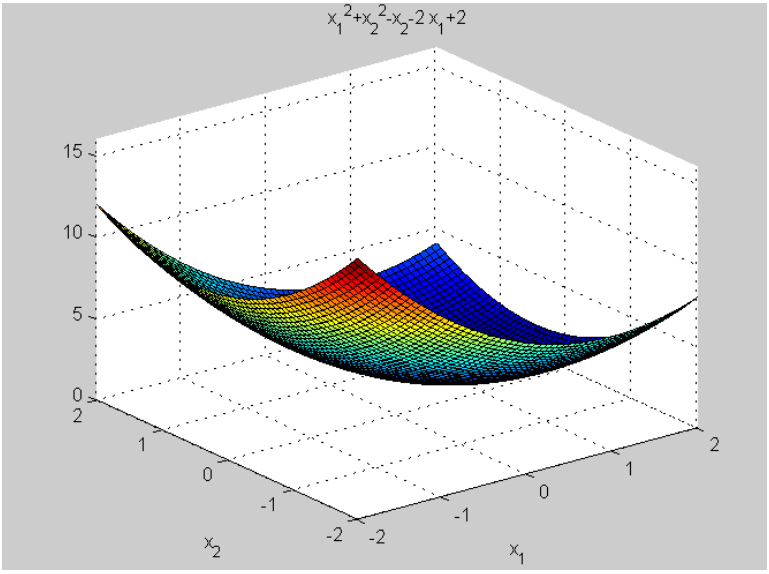


Рис. 2.6.1-1. Результат выполнения функции **ezsurf()**

Построим график (рис. 2.6.1-1) с использованием функции **ezsurf()**, аргументами которой служат: выражение функции, заключенное в одинарные кавычки, вектор изменения первой производной и вектор изменения второй переменной.

>> ezsurf ('x1.^2+x2.^2-x2-2*x1+2', [-2 2], [-2 2])

При нахождении минимума функции оформим многомерную функцию в виде m-функции (рис. 2.6.1-2).

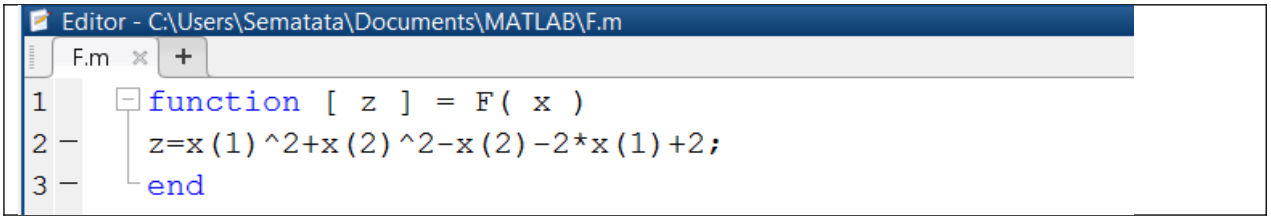


Рис. 2.6.1-2. Целевая функция **F(x)**

Определим координаты точки минимума и значение функции в этой точке с использованием функции Matlab **minsearch()** (рис. 2.6.1-3).

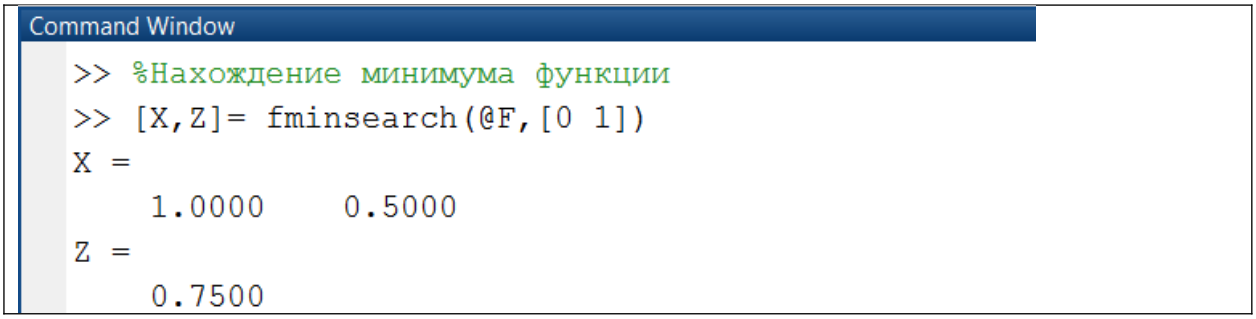


Рис. 2.6.1-3. Использование функции **fminsearch()** для нахождения минимума многомерной функции

2.6.2. Лабораторная работа по теме «Технология решения задач многомерной оптимизации»

1. Вопросы, подлежащие изучению

1) Построение графиков функции от двух переменных $\mathbf{F(x_1,x_2)}$ средствами пакета Matlab.
2) Нахождение координат точки минимум функции $\mathbf{F(x_1,x_2)}$ с использованием встроенных функций пакета Matlab - **fminsearch().**

2. Общее задание

1) ***Изучите*** *материал* ***Темы 2.6 (п. 2.6.1)****.*
2) ***Выберите*** *индивидуальное задание из* ***табл. 2.6.2-1****.*
3) ***Постройте*** *график функции* $\boldsymbol{F(x_1,x_2)}$**.**
4) ***Найдите*** *координаты точки минимум функции* $\boldsymbol{F(x_1,x_2)}$ *с использованием встроенной функции* ***fminsearch().***
5) ***Получите*** *значение функции в точке минимума.*
6) ***Сохраните*** *текст рабочего окна на внешнем носителе.*
7) ***Представьте*** *результаты работы преподавателю,* ***ответьте*** *на поставленные вопросы.*
8) ***Выполните*** *команду* ***clear all.***
9) ***Оформите отчет*** *по выполненной работе.*

3. Варианты индивидуальных заданий

Таблица 2.6.2-1

№	Функции для вычисления минимума
1	$F(x_1,x_2)=2x_1^2+5x_2^2-3x_1x_2-2x_1-22$
2	$F(x_1,x_2)=2x_1^2+x_2^2+2x_1x_2+x_1+x_2$
3	$F(x_1,x_2)=3x_1^2+5x_2^2-3x_1x_2+x_1+13$
4	$F(x_1,x_2)=x_1^2+2x_2^2-x_1x_2-x_2$
5	$F(x_1,x_2)=5x_1^2+3x_2^2-4x_1x_2+2x_1-18$
6	$F(x_1,x_2)=2x_1^2+3x_2^2+x_1+3x_2+10$
7	$F(x_1,x_2)=5x_1^2+x_2^2-2x_1x_2+2x_2+21$
8	$F(x_1,x_2)=2x_1^2+3x_2^2-2x_1x_2+x_1+3$
9	$F(x_1,x_2)=x_1^2+5x_2^2+x_2^2-x_1+2x_2+10$

10	$F(x_1, x_2)=4x_{1^2}+3x_{2^2}-3x_2+8$
11	$F(x_1, x_2)=3x_{1^2}+6x_{2^2}-2x_1+5$
12	$F(x_1, x_2)=x_{1^2}+2x_{2^2}-x_1x_2+x_2+16$
13	$F(x_1, x_2)=x_{1^2}+2x_{2^2}-x_1x_2+x_2+16$
14	$F(x_1, x_2)=x_{1^2}+4x_{2^2}+2x_1x_2+11$
15	$F(x_1, x_2)=2x_{1^2}+x_{2^2}-x_1x_2+2x_1$
16	$F(x_1, x_2)=x_{1^2}+2x_{2^2}+2x_1+x_2+4$
17	$F(x_1, x_2)=3x_{1^2}+x_{2^2}+x_1x_2+2x_1+3x_2+15$
18	$F(x_1, x_2)=2x_{1^2}+x_{2^2}+2x_1x_2+x_1-12$
19	$F(x_1, x_2)=2x_{1^2}+x_{2^2}+x_1x_2+5x_1+18$
20	$F(x_1, x_2)=3x_{1^2}+2x_{2^2}+x_1+2x_2+9$
21	$F(x_1, x_2)=2x_{1^2}+x_{2^2}-x_1x_2+x_1+2x_2$
22	$F(x_1, x_2)=2x_{1^2}+3x_{2^2}-x_1+x_2+26$
23	$F(x_1, x_2)=x_{1^2}+5x_{2^2}-x_1x_2-3x_2+13$
24	$F(x_1, x_2)=4x_{1^2}+3x_{2^2}-2x_1+11$
25	$F(x_1, x_2)=3x_{1^2}+6x_{2^2}+2x_1x_2$
26	$F(x_1, x_2)=x_{1^2}+2x_{2^2}-x_1x_2+2x_1-x_2+4$
27	$F(x_1, x_2)=x_{1^2}+3x_{2^2}+2x_1+3x_2+12$
28	$F(x_1, x_2)=x_{1^2}+4x_{2^2}+x_1x_2+2x_1-12$
29	$F(x_1, x_2)=5x_{1^2}+x_{2^2}+2x_1x_2+x_1+16$
30	$F(x_1, x_2)=3x_{1^2}+2x_{2^2}+x_1x_2+x_1+2x_2+9$

4. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в окне Command Window, снабженный необходимыми комментариями.

2.6.3. Контрольные вопросы по теме

1) Какую функцию называют многомерной?
2) Что является достаточным условием существования минимума для многомерной функции?
3) Назначение функции **fminsearch(name, x0)** и ее параметров.
4) Можно ли с использованием функции **fminsearch()** вычислить локальный максимум?
5) Что служит результатом выполнения функции **fminsearch()?**
6) Какие средства Matlab используются для описания целевой функции и как это влияет на параметры функции **fminsearch()?**

Список литературы

1. Васильев А.Н. Matlab. Самоучитель. Практический подход. – СПб.: Наука и Техника, 2012. – 448 с.
2. Дьяконов В.П. Полный самоучитель. – М.: ДМК Пресс, 2012.
3. Половко А.М., Бутусов П.Н. MATLAB для студентов. – СПб.: БХВ-Петербург, 2005. – 320 с., ил.
4. Кетков Ю.Л., Кетков А.Ю., Шульц М.М. Matlab 6.x: программирование численных методов. – СПб.: БХВ-Петербург, 2004. – 672 с., ил.

План УМД на 2016/17 уч.г.

Виктор Николаевич Шакин
Татьяна Игоревна Семенова

Основы работы с математическим пакетом Matlab

Учебное пособие

Подписано в печать 2015г. Формат 60x90 1/16.
Объём 8,3 усл.п.л. Тираж экз. Изд. № . Заказ .
ООО «Брис-М». Москва, ул. Авиамоторная, д. 8а.