*Тема 3.1.Рабочая среда MatLab и простейшие вычисления в окне CommandWindow*

**3.1.1. Элементы основного окна MatLab**

**3.1.2. Окно панели CommandWindow и простейшие вычисления**

**3.1.3. Основные объекты системы MatLab**

**3.1.4. Работа с векторами и матрицами**

**3.1.5. Окна Workspace иОкно Command History**

**3.1.6. Простейшие средства программированиям в MatLab**

##### Элементы основного окнаMatLab

Интерфейс MatLab аналогичен интерфейсу других Windows-приложений.После запуска MatLab на экране дисплея появляется **основное окно** системы MatLab в стандартной конфигурации, показанное на рис.3.1-1. При этом система MatLabготова к проведению вычислений в **командном режиме (**в окне**CommandWindow).**

**Основное окно** системы MatLab - это обычное окно приложений **MicrosoftWindows**, поэтому его можно перемещать, изменять в размерах, открывать на весь экран.

В **основном окне** системы MatLab размещены:

* **панель главного меню,** которое является контекстно-зависимым;
* **главная панель инструментов**;
* **окно панели CommandWindow** (Окно Команд);
* **окно панели CommandHistory** (Окно Истории Команд);
* **окно панели Workspace (**ОкноРабочей Области);
* **окно панели CurrentDirectory**(Окно Текущего Каталога) – на рис. 3.1-1 показана вкладка для перехода в это окно;
* **кнопка Start**;
* **строка состояния**.



Рис. 3.1-1. **Основное окно** системы MatLab после запуска

Две панели **Workspace** и **CurrentDirectory** закрывают друг друга, и для активизации нужной панели необходимо щелкнуть по соответствующей вкладке. В стандартной конфигурации все три оконных панели, вписанные в основное окно MatLab (рис.3.1-1)**,** закреплены **(**поставлены на якорь), то есть они могут передвигаться вместе с основным окном и вместе с ним изменять свои размеры. Однако границы между этими окнами можно изменять, то есть каждое из них можно открепить (снять с якоря), и тогда оно может занимать автономную позицию на экране.

***Панель главного меню***системы MatLab расположена вверху основного окна и имеет шесть элементов, команды которых показаны на рис. 3.1-2, в предположении, что активным является окно панели **CommandWindow**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File**- | содержит команды главного меню для работы с файлами команд и файлами рабочей области MatLab, а также команды установки параметров системы **MatLab**; | | | |
| **Edit**- | содержит команды главного меню для редактирования и очистки содержимого панелей основного окна; | | | |
| **Debug**- | содержит команды главного меню для работы с отладчиком **m**-**файлов**; | | | |
| **Desktop**- | содержит команды главного меню для управления панелями основного окна MatLab; | | | |
| **Windows**- | содержит команды главного меню для перевода в активное состояние различных окон MatLab; | | | |
| **Help**- | содержит команды главного меню для доступа к элементам справочной системы MatLab. | | | |
|  |  |

а) b)

|  |  |
| --- | --- |
|  |  |

c) d) e)

|  |  |
| --- | --- |
|  |  |

f) g)

Рис. 3.1-2. Команды элементов главного меню системы MatLa**b**

***Элемент меню File***, представленный на рис. 3.1-2а, содержит команды, позволяющие осуществлять операции с файлами, которые можно разделить по функциональным признакам на шесть частей.

С помощью команды **New** можно открыть окно редактора для создания и редактирования, программных **m**-файлов - окно **Editor**, графическое окно **Figure**для ви­зуализации результатов вычислений**,** создать новую переменную**,** открыть окно динамической модели и графического интерфейса пользователя (**GUIDE** - GraphicUserInterfaceDevelopmentEnvironment). Команда **Open** открывает стандарт­ное диалоговое окно **MSWindows**, в котором можно выбрать необходимый файл. Ко­манда **CloseCommandWindow** закрывает командное окно.

Следующая группа команд позволяет импортировать данные (**ImportData...**) с диска в рабочую область системы MatLab, а также сохранять дан­ные, расположенные в рабочей области, на диске (**SaveWorkspaceas**...).

Следующие две команды - **SetPath** и **Preferences** - открывают диа­логовые окна, в которых можно настраивать список путей доступа к файлам и параметры работы системы MatLab соответственно. Список путей дос­тупа используется при поиске необходимого файла во время выполнения команды. С помощью диалогового окна **Preferences** на­страиваются параметры работы инструментальных средств системы MatLab: размер и цвет шрифта, цвет фона, формат вывода числовой информа­ции и т.д.

Команды **PrintSetup ..., Print... и PrintSelection...** служат для открытия диалоговых окон, в которых производится настройка страницы для печати, выбор принтера и передача на печать содержимого текущего окна панели.

Следующая часть меню содержит до четырех последних имен файлов, с которыми производилась работа в системе MatLab.

Последняя команда - **ExitMATLAB** - осуществляет закрытие системы.

***Элемент меню Edit,*** показанный на рис. 3.1-2b, содержит команды, которые позволяют: отменять (**Undo**) или возвращать (**Redo**) отмененное действие; об­мениваться информацией с буфером обмена (**Cut** - Вырезать в буфер, **Сору** - Копировать в буфер, **Paste** - Вставить из буфера, **PasteSpecial** - Специаль­ная вставка); полностью выделять содержимое текущего окна панели (**SelectAll**); удалять выделенный текст (**Delete**);осуществлять поиск фрагмента текста в текущем окне панели (**Find**) или осуществлять поиск необходимого файл (**FindFile**), а также очищать окно команд (**ClearCommandWindow)**, окно истории команд (**ClearCommandHis­tory)** и содержимое рабочей области (**ClearWorkspace**). Некоторые команды этого элемента меню могут обозначаться серым цветом, который свидетельствует о том, что команда в данный момент недоступна.

***Элемент меню Debug****,*показанный на рис. 3.1-2c, содержит команды для работы с **m**-файлами, а также для отладки и редактирования этих файлов.

***Элемент меню Desktop***, изображенном на рис. 3.1-2e, размещает команды, которые позволяют отображать, скрывать, отделять или прикреплять оконные панели к главному окну системы MatLab. Команда **UndockCommandWindow** отделяет окно панели от основного окна и размещает его в поле экрана. Команда **DesktopLayout** открывает подменю с дополнительным набором команд, предназначенные для отображения на рабочем столе предустановленных комбинаций оконных панелей. Команда **Default** возвращает внеш­ний вид рабочего стола, который принят по умолчанию (рис. 3.1-1). Данное расположение основного окна системы MatLab является наиболее используемым при комплексной работе с системой MatLab. Команда **CommandWindowOnly** удаляет с рабочего стола все панели, кроме **окна панели CommandWindow**. Данную особенность полезно использовать, если работа с системой MatLab осуществляется только с помощью командного окна. Команда **HistoryandCommandWindow** служит для отображения на рабочем столе двух панелей: окна панели**CommandHistory** - с левой стороны и окна панели**CommandWindow** - с правой стороны. Дан­ная комбинация инструментальных средств на рабочем столе является более предпочтительной, чем одно окно команд, так как позволяет с помощью ок­на предыстории просматривать ранее введенные команды и при необходи­мости повторно выполнять различные последовательности этих команд. Команда **AllToolbar** отображает в основном окне шесть панелей: **Command Window, Current Directory, Workspace, Command History, Help и Profiler.**

Следующие шесть команд (**Command Window, Command History, Current Directory, Workspace, Help и Profiler)** управляют отображением соот­ветствующих оконных панелей на экране.

Последние три команды – **Toolbar, ShortcutsToolbar** и **Titles** - позволяют разрешить или запретить отображение заголовков соответствующих элементов основного окна.

***Элемент меню Window*,** представленном на рис. 3.1-2f, располагает коман­дами, которые позволяют переключаться между различными оконными панелями.

***Элемент меню Help***, команды которого изображены на рис. 3.1-2g, обеспечивают доступ к справочной информации о системе MatLab и демонстрационным файлам по функциям этой системы. Кроме того, с помощью команды **WebResources** можно пе­рейти на соответствующую страницу сайта фирмы-разработчика системы MatLab**.**

***Инструментальная линейка*** *(****Toolbar****)*дает наиболее простой и удобный способ работы с системой MatLab. Кнопки панели инструментов имеют изображение, явно подсказывающее их назначение (рис. 3.1-1). Для отображения подсказки о функции кнопки необходимо переместить кур­сор на кнопку и немного подождать.

##### Окно панели CommandWindow и простейшие вычисления

 Система MatLab создана таким образом, что любые вычисления можно выполнять в режиме прямых вычислений, то есть без написания программы. Это превращает MatLab в необычайно мощный калькулятор, который способен производить не только обычные для калькуляторов вычисления (например, выполнять арифметические операции и вычислять элементарные функции), но и выполнять операции с векторами и матрицами, комплексными числами, рядами и полиномами и т.п. Можно почти мгновенно задать различные функции и построить их графики.

Командное окно панели **CommandWindow**используется для ввода команд и вывода результатов их выполнения. Работа с командным окном 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 - **>>,** которое свидетельствует о готовности ядра системы MatLab к обработке очередной команды.

Таким образом, работа с системой в режиме прямых вычислений носит диалоговый характер и происходит по правилу «задал вопрос, получил ответ». То есть, пользователь набирает на клавиатуре, например, выражение, которое необходимо вычислить, редактирует его (если необходимо) в командной строке и завершает ввод нажатием клавиши <Enter>.

Рассмотрим простейший пример. Во-первых, вычислим результат выражения **2+3**. Для вычисления **2+3** необходимо ввести с клавиатуры в **Command Window.**

|  |
| --- |
| **Пример 3.1-1а** |
| **>>2+3** |

и нажать клавишу<Enter>**.**В итоге на следующей строке будет выведено:

|  |
| --- |
| **Пример 3.1-1b** |
| **ans =**  **5**  **>>** |

Далее о нажатии клавиши <Enter> упоминать не будем.

Из примера 3.1-1 видно, что MatLab по умолчанию создаёт переменную с именем **ans**, в которую записывает значение результата предыдущей операции (ее можно увидеть в рабочей области).

Рассмотрим второй пример.

|  |
| --- |
| **Пример 3.1-2а** |
| **>>a=sin(pi/2)**  **a =**  **1**  **>>** |

В этом примере создаётся переменная **а**, которой присваивается значение выражения **sin(pi/2)**, где **pi** – предопределённая в MatLab константа(существует множество других предопределённых констант, например **е**, **i -** мнимая единица и др.).

Если после выражения поставить точку с запятой, то результат вычисления не будет выведен на экран, но переменная**а** будет создана и ей будет присвоено вычисленное значение:

|  |
| --- |
| **Пример 3.1-2b** |
| **>>a=sin(pi/2);**  **>>** |

Значение этой переменной можно узнать, дважды щёлкнув по ней в рабочей области (**Workspace**), либо просто набрав её имя в командной строке.

|  |
| --- |
| **Пример 3.1-2c** |
| **>>a**  **a=**  **1**  **>>** |

Переменную, которой ранее было присвоено значение, можно использовать для дальнейших вычислений, например:

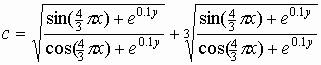
|  |
| --- |
| **Пример 3.1-2d** |
| **>> x= 3 - a**  **x =**  **2**  **>>** |

Если в выражении указан операнд, значение которого неизвестно**,** MatLabвыдает сообщение об ошибке:

|  |
| --- |
| **Пример 3.1-2е** |
| **>> (y+a)\*(y-a)**  **??? Undefined function or variable ‘y’.**  **>>** |

Если команда не помещается полностью в видимой части одной строки экрана, необходимо поставить три точки подряд, а затем нажать <Enter**>** и продолжать ввод команды на следующей строке.

Пусть, например, требуется найти значение выражения при x = 0.2 иy = –3.9**:**



Если набирать сразу все выражение, то получается достаточно длинная строка. Для переноса на следующую строку любой команды MatLab можно использовать знак переноса в виде трех подряд идущих точек. Вслед за знаком переноса необходимо нажать<Enter>**:**

|  |
| --- |
| **Пример 3.1-3а** |
| **>> x=0.2;**  **>> y=-3.9;**  **>> c=sqrt((sin(4/3\*pi\*x)+exp(0.1\*y))/(cos(4/3\*pi\*x)+exp(0.1\*y)))+...**  **((sin(4/3\*pi\*x)+exp(0.1\*y))/(cos(4/3\*pi\*x)+exp(0.1\*y)))^(1/3)**  **c =**  **2.0451**  **>>** |

Однако проще всего решить эту задачу, используя промежуточные переменные.

|  |
| --- |
| **Пример 3.1-3b** |
| **>> x=0.2;**  **>> y=-3.9;**  **>> a=sin(4/3\*pi\*x)+exp(0.1\*y);**  **>> b=cos(4/3\*pi\*x)+exp(0.1\*y);**  **>> c=sqrt(a/b)+(a/b)^(1/3)**  **c =**  **2.0451**  **>>** |

Здесь необходимо обратить внимание на некоторые важные особенности. Например, все операторы присваивания, кроме последнего, завершаются точкой с запятой для подавления вывода результата. Необязательно набирать выражение для **b**, похожее на только что введенное для **a**. После ввода третьей строки необходимо нажать клавишу **<↑>**. В командной строке появится предыдущее выражение, в которое следует внести необходимые изменения, а именно, необходимо заменить **sin()** на **cos(),b** на **a** и нажать<Enter>**.** Клавиши **<↑>** и **<↓>** служат для перехода по истории команд, то есть для занесения ранее набранных команд в командную строку. Для быстрого перехода по истории команд можно также использовать окно панели **CommandHistory**.

Необходимо обратить внимание, что в любой момент можно вывести значение переменной в командное окно, для чего следует набрать имя переменной в командной строке и нажать **<Enter>,** либо вызвать функцию **disp()**, например **disp(b)**.

Все переменные системы размещаются в рабочей области, содержимое которой (имена, размерность, тип) можно просмотреть командами **who**и **whos**.

Для очистки командного окна достаточно выполнить команду **clс,** а для очистки рабочей области – команду **clear.**

Необходимо иметь в виду, что любой фрагмент окна командной строки можно выделить и копировать в буфер, например, для переноса в Word или в командную строку. Возможен также перенос в командную строку текстовых фрагментов из других систем.

Все числовые значения, с которыми оперирует MatLab**,** в оперативной памяти компьютера представлены вещественными или комплексными значениями **double**. Это означает, что каждое вещественное число занимает **8 байт** в памяти и принимает по модулю значения из диапазона [**10-308;10+308**], причем количество значащих десятичных цифр достигает 16. Именно с такой точностью MatLab выполняет все вычисления.

Однако, при отображении числовых результатов на экране дисплея часть значащих цифр могут не отображаться в соответствии с установленным форматом вывода, причем отображаемые значения округляются по общепринятым в математике правилам.

Поскольку по умолчанию все вычисления в MatLabвыполняются с двойной точностью, формат вывода может быть установлен двумя способами: как программным путем с помощью команды **format** в окне панели **Command Window**, так и с помощью установки соответствующих **свойств окна панели CommandWindow.**

Для установки **свойств среды** системы MatLab**,** а конкретно **свойств окна панели CommandWindow,** необходимо активизировать элемент основного меню **File,**а затем выбрать команду **Preference,** а в раскрывшемся диалоговом окне **Preferences** выделить вкладку **Command Window** (рис. 3.1-3).

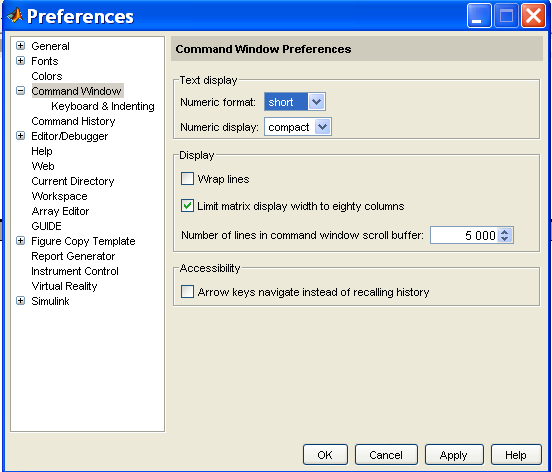


Рис. 3.1-3. Окно установки свойств среды системы MatLab

Внутри группы **Textdisplay** окна **Preferences** будут расположены раскрывающиеся списки **Numeric format** и **Numeric display.** Далее из раскрывающегося списка **Numeric format** можно установить один из форматов, приведенных в табл. 3.1-1.

Таблица 3.1-1

|  |  |
| --- | --- |
| **Формат** | **Описание** |
| **short** | Короткое число с плавающей точ­кой. Представляется с помощью четырех цифр после десятичной точки ( по умолчанию) |
| **long** | Длинное число с плавающей точ­кой. Представляется с помощью четырнадцати цифр после десятичной точки |
| **short e** | Короткое число с плавающей точ­кой. Представляется с помощью пяти разрядов, четыре из которых отводится под вывод дробной части |
| **long e** | Длинное число с плавающей точ­кой. Представляется с помощью шестнадцати разрядов, пятнадцать из которых отводится под вывод дробной части |
| **short g** | Выбирается наилучшая форма представления числа из форматов short и short e |
| **long g** | Выбирается наилучшая форма представления числа из форматов long и long e |
| **hex** | Число выводится в шестнадцатеричной форме |
| **+** | Символьное обозначение числа: «+» - положительное число; «-» - отрицательное число; пробел - нулевое значение |
| **rational** | Число выводится в дробном виде |

Для выделения результата вычисления или значения переменной MatLab вставляет пустую строку перед выводимым значением. Управлять появлением пустой строки или ее отсутствием можно в в раскрывшемся диалоговом окне **Preferences** внутри группы **Text display** с помощью следующих форматов:

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

Команда **format** служит для установки формата из командной строки. Например, обращение

|  |
| --- |
| **Пример 3.1-4** |
| **>>formatshort e**  **>>** |

аналогично выбору короткого формата в окне **Preferences**.

Еще раз необходимо напомнить, что вне зависимости от установленного формата все вычисления производятся с двойной точностью, следовательно, после смены формата с **short** на **long** не требуется повторно находить значения переменных. Достаточно снова вывести их значения в командном окне.

Также еще раз напомним, что содержимое командной строки **MatLab**легко редактируется. Курсор можно перемещать с помощью стрелок **<*←>, <→>*** и удалять неправильно набранные символы с помощью клавиш <Backspace**>** или **<**Delete**>.** Как было уже сказано, удобным свойством системы является возможность использовать клавиши-стрелки **<*↑>,***<↓> для доступа к стеку с ранее введенными командами. Таким образом, имеется возможность заново вызывать ранее вызванную команду, отредактировать ее и снова выполнить. Для небольших процедур это гораздо удобнее, чем писать и отлаживать специальные **m-файлы**, что требует постоянного перехода из окна MatLabв окно текстового редактора.

Все команды строчного редактора перечислены в таблице 3.1-2.

Таблица 3.1-2

|  |  |
| --- | --- |
| **Комбинация**  **клавиш** | **Назначение** |
| **→** | Перемещение курсора вправо на один символ |
| **←** | Перемещение курсора влево на один символ |
| **Ctrl+→** | Перемещение курсора вправо на одно слово |
| **Ctrl+←** | Перемещение курсора влево на одно слово |
| **Home** | Перемещение курсора в начало строки |
| **End** | Перемещение курсора в конец строки |
| **↑и ↓** | Перелистывание предыдущих команд вверх или вниз для подстановки в строку ввода |
| **Del** | Стирание символа справа от курсора |
| **Backspace** | Стирание символа слева от курсора |
| **Ctrl+k** | Стирание до конца строки |
| **Esc** | Очистка строки ввода |
| **Ins** | Вкл/выкл режима вставки |
| **PgUp** | Перелистывание страниц сессии вверх |
| **PgDn** | Перелистывание страниц сессии вниз |

##### Основные объекты системы MatLab

***Математические выражения*** в MatLab стоятся, как и в большинстве языков программирования, из числовых констант, переменных, стандартных и нестандартных функций, соединенных знаками арифметических операций, например **+**, **-** , **\*** , **/** , **^**, и круглых скобок. Кроме того, как было показано ранее, вид результата зависит от установленного формата.

***Число*** – простейший 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.**

Числа могут быть как **действительными**, так и **комплексными.** Комплексные числа содержат как действительные, так и комплексные чисти. В MatLab мнимая часть имеет множитель**i**или**j,** означающий корень квадратный из**-1.** Например,

**3i, 5j, -5.1 +i8, 005e-5- j0.006.**

Все операции над числами в MatLab выполняются в **формате с двойной точностью.** Однако, как нам уже известно, MatLab выдает числовые результаты в нормализованной форме с четырьмя цифрами после десятичной точки и одной до нее. Поэтому при работе с числовыми данными можно задавать различные форматы представления чисел.

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

***Системные константы*** (табл. 3.1-3) – это такие константы, значения которых задаются системой при загрузке. Однако по мере необходимости эти константы могут переопределяться.

Таблица 2.3.1-3 Таблица 3.5.3-3

|  |  |
| --- | --- |
| **Константа** | **Назначение** |
| **iили j** | Мнимая единица |
| **pi** | Число π=3.1415926… |
| **eps** | Погрешность вычислений над числами с плавающей точкой (по умолчанию 2-52) |
| **realmin** | Наименьшее число с плавающей точкой (по умолчанию 2-1022) |
| **realmax** | Наибольшее число с плавающей точкой (по умолчанию 21022) |
| **inf** | Значение машинной бесконечности |
| **NaN** | Указание на нечисловой характер данных (Not-a-Number) |

***Символьная константа*** – это последовательность символов, заключенных в одиночные апострофы. Например:

|  |
| --- |
| **'Кафедра ВМиП'**  **'88+0'** |

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

|  |
| --- |
| **% Это комментарий** |

***Переменные*** – это имеющие имена объекты, способные хранить некоторые, разные по значению, данные. В зависимости от этих данных, переменные могут быть числовыми или символьными, векторными или матричными. При использовании переменных необходимо придерживаться правил:

* имя переменной (ее идентификатор) может состоять из символов латинского алфавита, знака подчёркивания и цифр, но начинаться обязательно с символа алфавита;
* прописные и строчные буквы различаются;
* пробел не входит в имя переменной.

Для задания переменным определенных значений используется операция ***присваивания***, обозначаемая знаком равенства **=**.

|  |
| --- |
| **Пример 3.1-5** |
| >>a=3.25\*(0.7-3.3/5.1)+2.3^3  a =  12.3391  >>b=5\*(2.2+3.9i)+0.8  b =  11.8000 +19.5000i  >> |

Простейшие арифметические операции системы приведены втабл. 3.1-4.

Таблица 3.1-4

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Название | Операция | Синта-сис |
| **plus** | Плюс | **+** | **M1+M2** |
| **uplus** | Унарный плюс | **+** | **+M** |
| **minus** | Минус | **-** | **М1-M2** |
| **uminus** | Унарный минус | **-** | **-М** |
| **mtimes** | Матричное умножение | **\*** | **M1\*M2** |
| **times** | Поэлементное умножение массивов | **.\*** | **A1.\*A2** |
| **mpower** | Возведение матрицы в степень | **^** | **М^х** |
| **power** | Поэлементное возведение массива в степень | **.^** | **A.^x** |
| **mldivide** | Обратное (справа налево) деление матриц | **\** | **M1\M2** |
| **mrdivide** | Деление матриц слева направо | **/** | **M1/M2** |
| **ldivide** | Поэлементное деление массивов справа налево | **.\** | **A1.\A2** |
| **rdivide** | Поэлементное деление массивов слево направо | **./** | **A1./A2** |

Необходимо обратить внимание на то, что каждая операция имеет аналогичную по назначению функцию. Например, операции матричного умножения **\*** соответствует функция **mtimes(Ml,M2)**. Примеры применения арифметических операций уже не раз приводились, так что ограничимся несколькими дополнительными примерами.

|  |
| --- |
| **Пример3.1-6** |
| **>>А=[1 2 3];**  **>>В=[4 5 6];**  **>>В-А**  **ans=**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **3** | | **3                 3** | | |  | |  | **>>minus (В. А)** | | |  | |  | |  | **ans =** | | |  | |  | |  | **3** | | **3                 3** | | |  | |  | **>>А. ^ 2** | | |  | |  | |  | **ans =** | | |  | |  | |  | **1** | | **4                  9** | | |  | |  | **>>power(A,2)** | | |  | |  | |  | **ans =** | | |  | |  | |  | **1** | | **4          9** | | |  | |  | **>>А.\B** | | |  | |  | |  | **ans=** | | |  | |  | |  | **4.0000** | | **2.5000        2.0000** | | |  | |  | **>>ldivide(A,B)** | | |  | |  | |  | **ans=** | | |  | |  | |  | **4.0000** | | **2.5000        2.0000** | | |  | |  | **>>rdivide(A,B)** | | |  | |  | |  | **ans=** | | |  | |  | |  | **0.2500** | | **0.4000             0.5000** | | |  | |

Соответствие функций операторам и командам в системе MatLab является одним из основных положений программирования в MatLab. Оно позволяет одновременно использовать элементы как операторного, так и функционального программирования.

Следует отметить, что в математических выражениях операции имеют определенный **приоритет исполнения***.* Например, в MatLab приоритет логических операций выше, чем арифметических, приоритет возведения в степень выше приоритетов умножения и деления, приоритет умножения и деления выше приоритета сложения и вычитания. Для изменения приоритета операций в математических выражениях используются круглые скобки.

MatLab обладает большим набором ***стандартных встроенных математических функций***. Некоторые из них приведены в табл. 3.1-5. При вызове математических функций аргумент заключается в круглые скобки. Полный список всех встроенных элементарных математических функций можно получить, набрав в командной строке.

|  |
| --- |
| **Пример 3.1-7** |
| **>>help elfun**  **>>** |

Таблица 3.1-5

|  |  |
| --- | --- |
| ***Тригонометрические функции (результат вычисляется в радианах)*** | |
| **sin, cos, tan, cot** | Синус, косинус, тангенс и котангенс |
| **sec, csc** | Секанс, косеканс |
| ***Обратные тригономе-кие функции (результат вычисляется в рад.)*** | |
| **asin, acos,atan,atan2, acot** | Арксинус, арккосинус, арктангенс и арккотангенс |
| **asec, acsc** | Арксеканс, арккосеканс |
| ***Гиперболические функции*** | |
| **sinh, cosh, tanh, coth** | Гипербол. синус, косинус, тангенс и котангенс |
| **sech, csch** | Гиперболические секанс и косеканс |
| **asinh, acosh, atanh, acoth** | Гиперболические арксинус,  арккосинус, арктангенс и арккотангенс; |
| ***Экспоненциальная функция, логарифмы, степенные функции*** | |
| **exp** | Экспоненциальная функция |
| **log, log2, log10** | Логарифм натуральный, по основанию 2 и 10 |
| **sqrt** | Квадратный корень |
| ***Модуль, знак и функции для работы с комплексными числами*** | |
| **abs, sign** | Модуль и знак числа |
| **conj, imag, real** | Комплексно-сопряжённое |
| **mod, rem** | Остаток от деления с учетом знака делимого и без |
| **gcd, lcm** | Наибольший и наименьший общий делитель |
| **cell, fix, floor, round** | округления |

Команда **help** отображает в командном окне список разделов справочной системы. Для получения содержимого раздела необходимо указать через пробел его название после **help**, а для вывода детальной информации о какой-либо функции, следует ввести в строке с **help*имя функции***.

Возникающий в процессе вычислений комплексный результат не является ошибкой. MatLab автоматически переходит в область комплексных чисел, если это необходимо, продолжая вычисления. Например, если необходимо найти квадратный корень из **–1**. Более того, допустимы операции деления на ноль, которые приводят к стандартным переменным **Inf** или **–Inf**. Результат деления нуля на ноль есть **NaN** (Not a Number – не число). Переполнение или потеря точности в MatLab при выполнении операций с числами с плавающей точкой не вызывает прекращение вычислений.

Запись комплексных величин, используемых в арифметических выражениях, напоминает общепринятые математические стандарты. Мнимые части комплексных чисел сопровождаются либо буквой **i**, либо буквой **j**:

|  |
| --- |
| **Пример 3.1-8** |
| **>>x=1.5-0.5i**  **x =**  **1.5000 - 0.5000i**  **>>y=1.5-0.5j**  **y =**  **1.5000 – 0.5000j**  **>>** |

Если переменным **i** и **j** не присвоены какие-либо значения, то их можно использовать для формирования комплексных данных, используя знак умножения и располагая такой “сомножитель” до или после мнимой части:

|  |
| --- |
| **Пример3.1-9** |
| **>>a=1;**  **>>x=i\*a**  **x =**  **0 + 1.0000i**  **>>y=a\*i**  **y =**  **0 + 1.0000i**  **>>** |

Когда полные комплексные числа используются в операциях умножения, деления и возведения в степень, то для устранения неоднозначности их заключают в круглые скобки.

С помощью функции **real**и **imag** можно выделить вещественную и мнимую часть комплексного значения, а функция **complex** позволяет сконструировать комплексное значение по паре двух вещественных чисел.

|  |
| --- |
| **Пример3.1-10** |
| **>>b=5+j10;**  **>>c=real(d)**  **c=**  **5**  **>>d=imag(b)**  **d=**  **10**  **>>** |

Функцией **conj()** можно воспользоваться для получения комплексного сопряженного числа. Тот же результат можно получить, располагая апостроф вслед за комплексным значением.

|  |
| --- |
| **Пример3.1-11** |
| **>>t=conj(b)**  **t=**  **5.0000e+000 -1.0000e+001i**  **>>g=b'**  **g=**  **5.0000e+000 -1.0000e+001i**  **>>** |

Отметим, что над комплексными данными в MatLabопределены все арифметические операции, а комплексные операнды и выражения могут использоваться в качестве аргументов стандартных функций.

Необходимо обратить внимание, что числовые переменные в MatLab являются двумерными массивами размера один на один. Представление всех данных в MatLab в виде массивов оказывается очень полезным, о чем подробнее будет сказано в следующих разделах.

##### Работа с векторами и матрицами

Пакет MatLab построена как система, ориентирующая на работу с матрицами, то есть все численные вычисления производятся в матричной форме. Система MatLab выполняет сложные и трудоемкие операции над векторами и матрицами даже в режиме прямых вычислений без какого-либо программирования. Ею можно пользоваться как мощнейшим калькулятором, в котором наряду с обычными арифметическими и алгебраическими действиями могут использоваться такие сложные операции, как инвертирование матрицы, вычисление ее собственных значений и принадлежащих им векторов, решение систем линейных уравнений, вывод графиков двумерных и трехмерных функций и многое другое. Важно отметить, что даже обычные числа и переменные в MatLab рассматриваются как матрицы размера **1 x 1**, что дает единообразные формы и методы проведения операций над обычными числами и массивами. Это также означает, что большинство функций может работать с аргументами в виде векторов и матриц. При необходимости вектора и матрицы преобразуются в массивы, и значения вычисляются для каждого их элемента.

Массивы являются одним из самых распространенных способов хранения данных и используются во всех языках программирования и математических пакетах. К особенностям работы с массивами в MatLab относится то, что одномерный массив может быть **вектор-строкой** или **вектор-столбцом**. Если способ представления массива важен, то мы будем подчеркивать, о строке или о столбце идет речь. Если же это несущественно, то будем говорить о **вектор-строках** и **вектор-столбцах** просто как о векторах или одномерных массивах. Напомним, что одномерный массив в MatLab есть двумерный, у которого ода из размерностей равна единице.

Для определения вектора используются квадратные скобки, а элементы вектора отделяются друг от друга:

* точкой с запятой, если требуется получить вектор–столбец;
* пробелом или запятой, если необходимо разместить элементы в векторе-строке.

|  |
| --- |
| **Пример3.1-12** |
| **>>a=[0.2; -3.9; 4.6]**  **a =**  **0.2000**  **-3.9000**  **4.6000**  **>>b=[7.6; 0.1; 2.5]**  **b =**  **7.6000**  **0.1000**  **2.5000**  **>>u=[0.1 0.5 -3.7 8.1]**  **u =**  **0.1000 0.5000 -3.7000 8.1000**  **>>v=[5.2 9.7 3.4 –0.2]**  **v =**  **5.2000 9.7000 3.4000 –0.2000**  **>>** |

Получить информацию о переменных, как мы уже знаем, можно с помощью окна **Workspace** или при помощи команды **whos.**

Для определения длины вектора используется функция **length(а)**, вектор **а** указывается в качестве ее входного аргумента.

|  |
| --- |
| **Пример3.1-13** |
| **>>L=length(a)**  **L =**  **3**  **>>** |

**Вектор-столбцы** с одинаковым числом элементов можно складывать и вычитать друг из друга при помощи опрераций **"+" и "–".** Эти правила верны и для **вектор-строк**.

Сложение и вычитание **вектор-строк** и **вектор-столбцов** или векторов разных размеров приводит к ошибке. Операция **\*** предназначена для умножения векторов по правилу матричного умножения. Поскольку MatLab различает **вектор-строки** и **вектор- столбцы**, то допустимо либо умножение **вектор-строки** на такой же по длине **вектор-столбец** (**скалярное произведение**), либо умножение **вектор-столбца** на **вектор-строку** (**внешнее произведение**, в результате которого получается **прямоугольная матрица**). **Скалярное произведение** двух векторов возвращает функция **dot()**, а **векторное** - **cross()**:

|  |
| --- |
| **Пример3.1-14** |
| **>>s=dot(a, b)**  **s=**  **1.2630e+001**  **>>c=cross(a, b)**  **c=**  **-1.0210e+001**  **3.4460e+001**  **2.9660e+001**  **>>** |

Для операции транспонирования зарезервирован символ апостроф - **'**. Если вектор содержит комплексные числа, то операция  **'** приводит к комплексно-сопряженному вектору. При вычислении скалярного и векторного произведений функциями **cross()** и **dot()** не обязательно следить за тем, чтобы оба вектора были либо столбцами, либо строками. Результат получается верный, например, при обращении **c=cross(a,b')**, только **c** становится вектор-строкой.

MatLab поддерживает поэлементные операции с векторами. Наряду с умножением по правилу матричного умножения, существует операция поэлементного умножения - **.\*** (точка со звездочкой). Данная операция применяется к векторам одинаковой длины и приводит к вектору той же длины, что исходные, элементы которого равны произведениям соответствующих элементов исходных векторов. Например, для векторов **a** и **b**, введенных выше, поэлементное умножение дает следующий результат:

|  |
| --- |
| **Пример3.1-15** |
| **>>c=a.\* b**  **c =**  **1.5200**  **-0.3900**  **11.5000**  **>>** |

Аналогичным образом работает поэлементное деление - **./** (точка с косой чертой). Кроме того, операция **.\**(точка с обратной косой чертой) осуществляет обратное поэлементное деление, то есть выражения **a./b** и **b.\a** эквивалентны. Возведение элементов вектора **a** в степени, равные соответствующим элементам **b**, производится с использованием операции - **.^**. Для транспонирования **вектор-строк** или **вектор-столбцов** предназначено сочетание - **.'** (точка с апострофом). Операции **- '** и **.'** для вещественных векторов приводят к одинаковым результатам. Не обязательно применять поэлементные операции при умножении вектора на число и числа на вектор, делении вектора на число, сложении и вычитании вектора и числа. При выполнении, например, операции **a\*2**, результат представляет собой вектор того же размера, что и **a**, с удвоенными элементами.

Векторы могут быть аргументами встроенных математических функций, таких, как **sin(), cos()** и т. д. В результате получается вектор с элементами, равными значению вызываемой функции от соответствующих элементов исходного вектора, например:

|  |
| --- |
| **Пример3.1-16** |
| **>>q=sin([0 pi/2 pi])**  **q =**  **0 1.0000 0.0000**  **>>** |

Однако для вычисления более сложной функции от вектора значений, скажем, где α**,** например**,**  является вектором-строкой, состоящей из четырех элементов, выражение f=(α\*sin(α)+α^2)/(α+1) вызовет ошибку уже при попытке умножения αна sin(α**)**. Дело в том, что α является **вектор-строкой** длиной четыре, то есть хранится в двумерном массиве размером один на четыре. Точно также представлена и функция sin(α), следовательно, умножение при помощи звездочки (по правилу матричного умножения) лишено смысла. Аналогичная ситуация возникает и при возведении вектора α в квадрат, то есть, фактически, при вычислении α\*α**.**

Правильная запись выражения в MatLab требует использования поэлементных операций.

|  |
| --- |
| **Пример3.1-17** |
| **>>f=(a.\*sin(a)+a.^2)./(a+1);**  **>>** |

Часто требуется вычислить функцию от вектора значений аргумента, отличающихся друг от друга на постоянный шаг. Для создания таких **векторов-строк** предусмотрена операция двоеточие - **:**, которая отделяет начальное значение аргумента, шаг и конечное значение аргумента.

|  |
| --- |
| **Пример3.1-18** |
| **>>x=-1.2:0.5:1.8**  **x =**  **-1.2000 -0.7000 -0.2000 0.3000 0.8000 1.3000 1.8000**  **>>f=(x.\*sin(x)+x.^2)./(x+1)**  **f =**  **-12.7922 3.1365 0.0997 0.1374 0.6744 1.2794 1.7832**  **>>** |

Шаг может быть отрицательным, в этом случае начальное значение должно быть больше, либо равно конечному значению для получения непустого вектора. Если шаг равен единице, то его можно не указывать, например:

|  |
| --- |
| **Пример3.1-17** |
| **>>n=-3:4**  **n =**  **-3 -2 -1 0 1 2 3 4**  **>>** |

Ясно, что для заполнения вектор-столбца элементами с постоянным шагом следует транспонировать вектор-строку.

Создание векторов при помощи двоеточия и умение производить поэлементные операции необходимо для визуализации массивов данных.

|  |
| --- |
| **Пример3.1-18** |
| **>>x=10:-2:0**  **x =**  **10 8 6 4 2 0**  **>>sin(x)=**  **-0.5440 0.9894 -0.2794 -0.7568 0.9093 0**  **>>** |

Необходимо отметить, что при умножении векторов друг на друга получается скаляр.

|  |
| --- |
| **Пример 3.1-19** |
| **>>y=[2 3 5 8 3 9];**  **>>x\*y**  **??? Error using ==> \***  **Inner matrix dimensions must agree.**  **>>** |

Здесь допущена ошибка, так как матрицы (векторы) не согласованны. Правильно будет

|  |
| --- |
| **Пример3.1-20a** |
| **>>x\*y'**  **ans =**  **112**  **>>** |

или можно было создать **у** как матрицу-столбец:

|  |
| --- |
| **Пример3.1-20b** |
| **>>y=[2;3;5;8;3;9]**  **>>** |

MatLab обладает большим набором встроенных функций для обработки векторных данных, часть из них приведена в табл. 5.3-6.

Таблица. 3.1-6

|  |  |
| --- | --- |
| **Функции** | **Назначение** |
| **s=sum(a)** | Сумма всех элементов вектора **a** |
| **p=prod(a)** | Произведение всех элементов вектора **a** |
| **m=max(a)** | Нахождение максимального значения среди элементов вектора **a** |
| **[m,k]=max(a)** | Второй выходной аргумент **k** содержит номер максимального элемента в векторе **a** |
| **m=min(a)** | Нахождение минимального значения среди элементов вектора **a** |
| **[m,k]=min(a)** | Второй выходной аргумент **k** содержит номер минимального элемента в векторе **a** |
| **m=mean(a)** | Вычисление среднего арифметического элементов вектора **a** |
| **a1=sort(a)** | Упорядочение элементов вектора **a** по возрастанию |
| **[a1,ind]=sort(a)** | Второй выходной аргумент **ind** является вектором из целых чисел от **1** до **length(a)**, который соответствует проделанным перестановкам |

Полный список имеющихся функций выводится в командное окно при помощи команды **helpdatafun**, а для получения подробной информацию о каждой функции требуется указать ее имя в качестве аргумента команды **help.** Обратите внимание на то, что ряд функций допускает обращение к ним как с одним, так и с двумя выходными аргументами. В случае нескольких выходных аргументов они заключаются в квадратные скобки и отделяются запятой.

Очень часто требуется обработать только часть вектора, или обратиться к некоторым его элементам. Разберем правила MatLab, по которым производится индексация векторных данных. Для доступа к элементу вектора необходимо указать его номер в круглых скобках сразу после имени переменной, в которой содержится вектор. Например, сумма первого и третьего элементов вектора **y** находится при помощи выражения

|  |
| --- |
| **Пример3.1-21** |
| **>>s=y(1)+y(3);**  **>>** |

Обращение к последнему элементу вектора можно произвести с использованием аргумента **end,** то есть **y(end)** и **y(length(v))** приводят к одинаковым результатам.

Указание номеров элементов вектора можно использовать и при вводе векторов, последовательно добавляя новые элементы (не обязательно в порядке возрастания их номеров).

|  |
| --- |
| **Пример3.1-22** |
| **>>h=10;**  **>>h(2)=20;**  **>>h(4)=40**  **h =**  **10 20 0 40**  **>>** |

Заметим, что для ввода первого элемента **h** не обязательно указывать его индекс, так как при выполнении оператора **h=1** создается вектор (массив размера один на один). Следующие операторы присваивания приводят к автоматическому увеличению длины вектора **h**, а пропущенные элементы (в нашем случае **h(3)**) получают значение ноль.

Индексация двоеточием позволяет выделить идущие подряд элементы в новый вектор. Начальный и конечный номера указываются в круглых скобках через двоеточие, например:

|  |
| --- |
| **Пример3.1-23** |
| **>>z=[0.2 -3.8 7.9 4.5 7.2 -8.1 3.4];**  **>>znew=z(3:6)**  **znew =**  **7.9000 4.5000 7.2000 -8.1000**  **>>** |

Применение встроенных функций обработки данных к некоторым последовательно расположенным элементам вектора не представляет труда. Следующий вызов функции **prod()** вычисляет произведение элементов вектора **z**со второго по шестой:

|  |
| --- |
| **Пример3.1-24** |
| **>>p=prod(z(2:6))**  **>>** |

Индексация векторов служит для выделения элементов с заданными индексами в новый вектор. Индексный вектор должен содержать номера требуемых элементов, например:

|  |
| --- |
| **Пример.3.1-25** |
| **>>ind=[3 5 7];**  **>>znew=z(ind)**  **znew =**  **7.9000 7.2000 3.4000**  **>>** |

Для нахождения суммы элементов произвольного вектора **z** с четными индексами необходимо выполнить следующие команды:

|  |
| --- |
| **Пример3.1-26** |
| **>>ind=2:2:length(z);**  **>>s=sum(z(ind))**  **>>** |

К

Конструирование новых векторов из элементов имеющихся векторов производится при помощи квадратных скобок. Следующий оператор приводит к образованию вектора, в котором пропущен пятый элемент вектора **z.**

|  |
| --- |
| **Пример3.1-27** |
| **>>znew=[z(1:4) z(6:end)]**  **znew =**  **0.2000 -3.8000 7.9000 4.5000 -8.1000 3.4000**  **>>** |

Для определения матрицы необходимо задать значение элементов строк и разделить строки матрицы символом - **;**, заключенными в квадратные скобки:

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

Кроме того для определения матриц существует множество функций. Некоторые из них приведены в табл. 3.1-7.

Таблица. 3.1-7

|  |  |
| --- | --- |
| **Функция** | **Результат и примеры вызовов** |
| **zeros** | Нулевая матрица  **F=zeros(4,5) F=zeros(3) F=zeros([3 4])** |
| **eye** | Единичная прямоугольная матрица (единицы расположены на главной диагонали)  **I=eye(5,8) I=eye(5) I=eye([5 8])** |
| **ones** | Матрица, целиком состоящая из единиц  **E=ones(3,5) E=ones(6) E=ones([2 5])** |
| **rand** | Матрица, элементы которой — случайные числа, равномерно распределенные на интервале (0,1)  **R=rand(5,7) R=rand(6) R=rand([3 5])** |
| **randn** | Матрица, элементы которой — случайные числа, распределенные по нормальному закону с нулевым средним и дисперсией, равной единице N**=randn(5,3) N=randn(9) N=randn([2 4])** |
| **diag** | 1) диагональная матрица, элементы которой задаются во входном аргументе – векторе **D=diag(v)**  2) диагональная матрица со смещенной на **k** позиций диагональю (положительные k — смещение вверх, отрицательные — вниз), результатом является квадратная матрица размера **length(v)+abs(k) D=diag(v,k)**  3) выделение главной диагонали из матрицы в вектор **d=diag(A)**  4) выделение **k**-ой диагонали из матрицы в вектор **d=diag(A,k)** |

Рассмотрим примеры формирования матриц.

|  |
| --- |
| **Пример3.1-28** |
| **>>Z=zeros(3,5)**  **Z =**  **0 0 0 0 0**  **0 0 0 0 0**  **0 0 0 0 0**  **>>A=[1 2 3 4; 5 6 7 8; 9 10 11 12] ; %матрица А(4х3)**  **A =**  **1 2 3 4**  **5 6 7 8**  **9 10 11 12**  **>>** |

Причем **элементами** матрицы могут быть **матрицы**или**вектора**.

|  |
| --- |
| **Пример3.1-29** |
| **% первая "строка" матрицы К–матрица А,**  **% вторая – вектор [1 2 3 4]**  **>>K=[A;1 2 3]**  **K =**  **1 2 3 4**  **5 6 7 8**  **9 10 11 12**  **1 2 3 4**  **>>** |

Таким образом, как нам уже известно, вектор является матрицей с числом строк или столбцов равным **1,** а число – матрица из одного элемента:

|  |
| --- |
| **Пример3.1-30** |
| **>>x=[1 2 3 4]**  **x =**  **1 2 3 4**  **>>c=1.256**  **c = 1.2560**  **>>** |

Над матрицами и элементами матрицы можно выполнять различные операции, но при этом необходимо учитывать правила работы с матрицами. Рассмотрим примеры этих операций.

1. **Умножение матрицы A на матрицу B:**

|  |
| --- |
| **Пример.3.1-31** |
| **>>B=[2 5; 6 4; 6 5; 8 3]**  **B =**  **2 5**  **6 4**  **6 5**  **8 3**  **>>C=A\*B**  **C =**  **64 40**  **152 108**  **240 176**  **>>** |

1. **Умножение соответствующих элементов матриц необходимо использовать оператор поэлементного доступа ".":**

|  |
| --- |
| **Пример3.1-32** |
| **>>A1=[1 3 4 6; 9 7 4 0; 8 6 3 9];**  **>>С2=A.\*A1**  **С2 =**  **1 6 12 24**  **45 42 28 0**  **72 60 33 108**  **>>** |

1. **Добавление элементов** ("увеличение матрицы"):

|  |
| --- |
| **Пример3.1-33** |
| **>>a1=[A;x]**  **a1 =**  **1 2 3 4**  **5 6 7 8**  **9 10 11 12**  **1 2 3 4**  **>>** |

Пара квадратных скобок при этом является оператором объединения.

1. **Транспонирование матрицы (**апостроф - **' ):**

|  |
| --- |
| **Пример3.1-34** |
| **>>Bt=B'**  **Bt =**  **2 6 6 8**  **5 4 5 3**  **>>** |

1. **Удаление строки** (столбца матрицы), в примере удален второй столбец:

|  |
| --- |
| **Пример3.1-35** |
| **>>A(:,2)=[ ] %круглые скобки – оператор извлечения**  **A =**  **1 3 4**  **5 7 8**  **9 11 12**  **>>** |

1. **Извлечение элемента**матрицы:

|  |
| --- |
| **Пример3.1-36** |
| **>>y=A(3,3)**  **%обращение к элементу в 3-й строке и 3-м столбце**  **y =**  **12**  **>> % или**  **>> A(5) %обращение по номеру так,**  **%если записать все столбцы матрицы друг за другом>**  **ans = 7** |

**7) Извлечение строк или столбцов** матрицы:

Для извлечения строк или столбцов вместо номера элементов в строке или столбце используется двоеточие, которое является оператором перечисления (с первого до последнего элемента строки или столбца):

|  |
| --- |
| **Пример3.1-37** |
| **>>%извлечение 3-й строки из А,**  **>>%двоеточие означает весь набор чисел**  **>>y=A(3,:)**  **y =**  **9 11 12**  **>>% в этом примере это элементы из 3-й стоки и 1 2 3 столбцов.** |

Двоеточие, поставленное между двумя числами – это перечисление от одного до другого числа, с указанным интервалом (по умолчанию **1**):

|  |
| --- |
| **Пример3.1-38** |
| **>>x=0:5 %вектор х = (0 1 2 3 4 5)**  **x =**  **0 1 2 3 4 5**  **>>x=0:0.1:0.5 %вектор х = (0 0,2 0,3 0,4 0,5)**  **x =**  **0 0.1000 0.2000 0.3000 0.4000 0.5000**  **>>** |

Таким образом, матрицы небольших размеров удобно вводить из командной строки. Существует четыре способа ввода матриц. Например, необходимо определить следующую матрицу:



***Первый способ*** предполагает набрать в командной строке (разделяя элементы строки матрицы пробелами): **A=[0.7 –2.5 9.1** и нажать **<Enter>**. Курсор перемещается в следующую строку (символ приглашения командной строки **>>** отсутствует). Элементы каждой следующей строки матрицы набираются через пробел, а ввод строки завершается нажатием на <Enter>. При вводе последней строки в конце ставится закрывающая квадратная скобка:

|  |
| --- |
| **Пример3.1-39a** |
| **>>A=[0.7 -2.5 9.1**  **8.4 0.3 1.7**  **-3.5 6.2 4.7]**  **A =**  **0.7000 -2.5000 9.1000**  **8.4000 0.3000 1.7000**  **-3.5000 6.2000 4.7000**  **>>** |

***Второй способ*** ввода матрицы основан на том, что матрицу можно рассматривать как вектор-столбец, каждый элемент которого является строкой матрицы. Поскольку точка с запятой используется для разделения элементов вектора-столбца, то ввод, к примеру, матрицы

осуществляется оператором присваивания:



|  |
| --- |
| **Пример2.3.1-39b** |
| **>>B=[6.1 0.3; -7.9 4.4; 2.5 -8.1]**  **B =**  **6.1000 0.3000**  **-7.9000 4.4000**  **2.5000 -8.1000**  **>>** |

***Третий способ*** Очевидно, что допустима такая трактовка матрицы, при которой она считается вектор-строкой, каждый элемент которой является столбцом матрицы. Следовательно, для ввода матрицы

достаточно воспользоваться командой:



|  |
| --- |
| **Пример3.1-40** |
| **>>C=[[0.4; 0.1] [-7.2; -2.1] [5.3; -9.5]]**  **C =**  **0.4000 -7.2000 5.3000**  **0.1000 -2.1000 -9.5000**  **>>** |

Еще можно воспользоваться командой **whos** для получения информации о переменных **A, B** и **C** рабочей среды. В командное окно выводится таблица с информацией о размерах массивов, памяти, необходимой для хранения каждого из массивов, и типе – **doublearray**:

|  |
| --- |
| **Пример3.1-41** |
| **>>whos A B C**  **Name Size Bytes Class**  **A 3x3 72 double array**  **B 3x2 48 double array**  **C 2x3 48 doublearray**  **>>** |

Функция **size()**позволяет установить размеры массивов, она возвращает результат в виде вектора, первый элемент которого равен числу строк, а второй – столбцов:

|  |
| --- |
| **Пример3.1-42** |
| **>>s=size(B)**  **s =**  **3 2**  **>>** |

Сложение и вычитание матриц одинаковых размеров производится с использованием знаков **+, -.** Звездочка **\*** служит для вычисления матричного произведения, причем соответствующие размеры матриц должны совпадать.

Допустимо умножение матрицы на число и числа на матрицу, при этом происходит умножение каждого элемента матрицы на число и результатом является матрица тех же размеров, что и исходная. Апостроф - **'** предназначен для транспонирования вещественной матрицы или нахождения сопряженной к комплексной матрице. Для возведения квадратной матрицы в степень применяется знак **^**.

MatLab обладает многообразием различных функций и способов для работы с матричными данными. Для обращения к элементу двумерного массива следует указать его номер строки и номер столбца в круглых скобках после имени массива.

Индексация двоеточием позволяет получить часть матрицы — строку, столбец или блок, например:

|  |
| --- |
| **Пример3.1-43** |
| **>>c1=A(2:3,2)**  **c1 =**  **0.3000**  **6.2000**  **>>r1=A(1,1:3)**  **r1 =**  **0.7000 -2.5000 9.1000**  **>>** |

Для обращения ко всей строке или всему столбцу не обязательно указывать через двоеточие начальный (первый) и конечный индексы, то есть операторы **r1=A(1,1:3)** и **r1=A(1,:)** эквивалентны. Для доступа к элементам строки или столбца от заданного до последнего можно использовать **end,** так же как и для векторов: **A(1,2:end)**. Выделение блока, состоящего из нескольких строк и столбцов, требует индексации двоеточием, как по первому измерению, так и по второму.

Пусть в массиве **T** хранится матрица:



Для выделения элементов матрицы **T**со второй строки по третью и со второго столбца по четвертый, достаточно использовать оператор:

|  |
| --- |
| **Пример3.1-44** |
| **>>c1=A(2:3,2)**  **c1 =**  **0.3000**  **6.2000**  **>>** |

Ранее было описано применение поэлементных операций к векторам. Поэлементные вычисления с матрицами производятся практически аналогично, разумеется, необходимо следить за совпадением размеров матриц:

* **A.\*B, A./B** - поэлементные умножение и деление;
* **A.^p** - поэлементное возведение в степень, **p** - число;
* **A.^B** - возведение элементов матрицы **A** в степени, равные соответствующим элементам матрицы **B**;
* **A.'** - транспонирование матрицы (для вещественных матриц **A'** и **A.'** приводят к одинаковым результатам).

Иногда требуется не просто транспонировать матрицу, но и "развернуть" ее. Разворот матрицы на **90o** против часовой стрелки осуществляет функция **rot90( )**:

|  |
| --- |
| **Пример3.1-45** |
| **>>Q=[1 2;3 4]**  **Q =**  **1 2**  **3 4**  **>>R=rot90(Q)**  **R =**  **2 4**  **1 3**  **>>** |

Допустимо записывать сумму и разность матрицы и числа, при этом сложение или вычитание применяется, соответственно, ко всем элементам матрицы. Вызов математической функции от матрицы приводит к матрице того же размера, на соответствующих позициях которой стоят значения функции от элементов исходной матрицы.

В MatLab определены и матричные функции, например, **sqrtm( )** предназначена для вычисления квадратного корня. Например, найдем квадратный корень из матрицы и проверим полученный результат, возведя его в квадрат:



|  |
| --- |
| **Пример2.3.1-46** |
| **>>K=[3 2; 1 4];**  **>> S=sqrtm(K)**  **S =**  **1.6882 0.5479**  **0.2740 1.9621**  **>>S\*S**  **ans =**  **3.0000 2.0000**  **1.0000 4.0000**  **>>** |

Матричная экспонента вычисляется с использованием **expm()**. Специальная функция **funm()**служит для вычисления произвольной матричной функции.

Все функции обработки данных могут быть применены и к двумерным массивам. Основное отличие от обработки векторных данных состоит в том, что эти функции работают с двумерными массивами по столбцам, например, функция **sum()** суммирует элементы каждого из столбцов и возвращает вектор-строку, длина, которой равна числу столбцов исходной матрицы:

|  |
| --- |
| **Пример3.1-47** |
| **>>M=[1 2 3; 4 5 6; 7 8 9]**  **M =**  **1 2 3**  **4 5 6**  **7 8 9**  **>>s=sum(M)**  **s =12 15 18** |

Если в качестве второго входного аргумента **sum()** указать **2**, то суммирование произойдет по строкам. Для вычисления суммы всех элементов матрицы требуется дважды применить **sum()**:

|  |
| --- |
| **Пример3.1-48** |
| **>>s=sum(sum(M))**  **s =45**  **>>** |

Очень удобной возможностью MatLab является конструирование матрицы из матриц меньших размеров. Пусть заданы матрицы:



Требуется составить из **M1, M2, M3**и **M4** блочную матрицу **M**

.



Можно считать, что **M**имеет размеры два на два, а каждый элемент является, соответственно, матрицей **M1, M2, M3**или **M4**. Следовательно, для получения в рабочей среде **MatLab** матрицы **M** требуется использовать оператор:

|  |
| --- |
| **Пример2.3.1-49** |
| **>>M=[M1 M2; M3 M4];**  **>>** |

**Четвертый способ**. 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 **ArrayEditor.**

Окно **ArrayEditor** (окно редактора массива данных) состоит из панели инструментов и области просмотра значений переменных. В случае открепленного от рабочего стола окна редактирования данных в окне присутствует главное меню, которое совпадает с главным меню рабочего стола, и строка состояния (рис. 3.1-4). В окне редактора данных можно отображать не­сколько переменных. Переключение между переменными реализуется **ArrayEditor**. На   
рис. **3.1**-4 в окне редактирования данных находится переменная **d**.

Панель инструментов окна редактирования данных используется для записи данных (**Save**), для пе­ремещения (**Cut**) и копирования *(***Сору**) в буфер обмена выделенных значений, вставки (**Paste**) значений из буфера и печати (**Print**).При перемещении выде­ленных значений в буфер обмена на их месте будут записаны нули. Количество и размерность числовых значений при вставке из буфера обмена должны сов­падать с областью, выделенной для вставки.



Рис**3.1**-4. Окно редактора данных

##### Окна Workspace иОкно Command History

Как было сказано ранее, основное меню является контекстно-зависимым. Поэтому при активном окне **Workspace**элементы основного меню (рис. **3.1**-5) будут отличаться от элементов при активном окне рабочей области.

Как видно из рис.**2.3.1**-5 здесь появились дополнительные элементы:

|  |  |
| --- | --- |
| **View**- | содержит команды главного меню для отображения в окне **Workspace** различной информации и ее сортировки; |
| **Graphics**- | содержит команды главного меню для работы с графическим окном. |

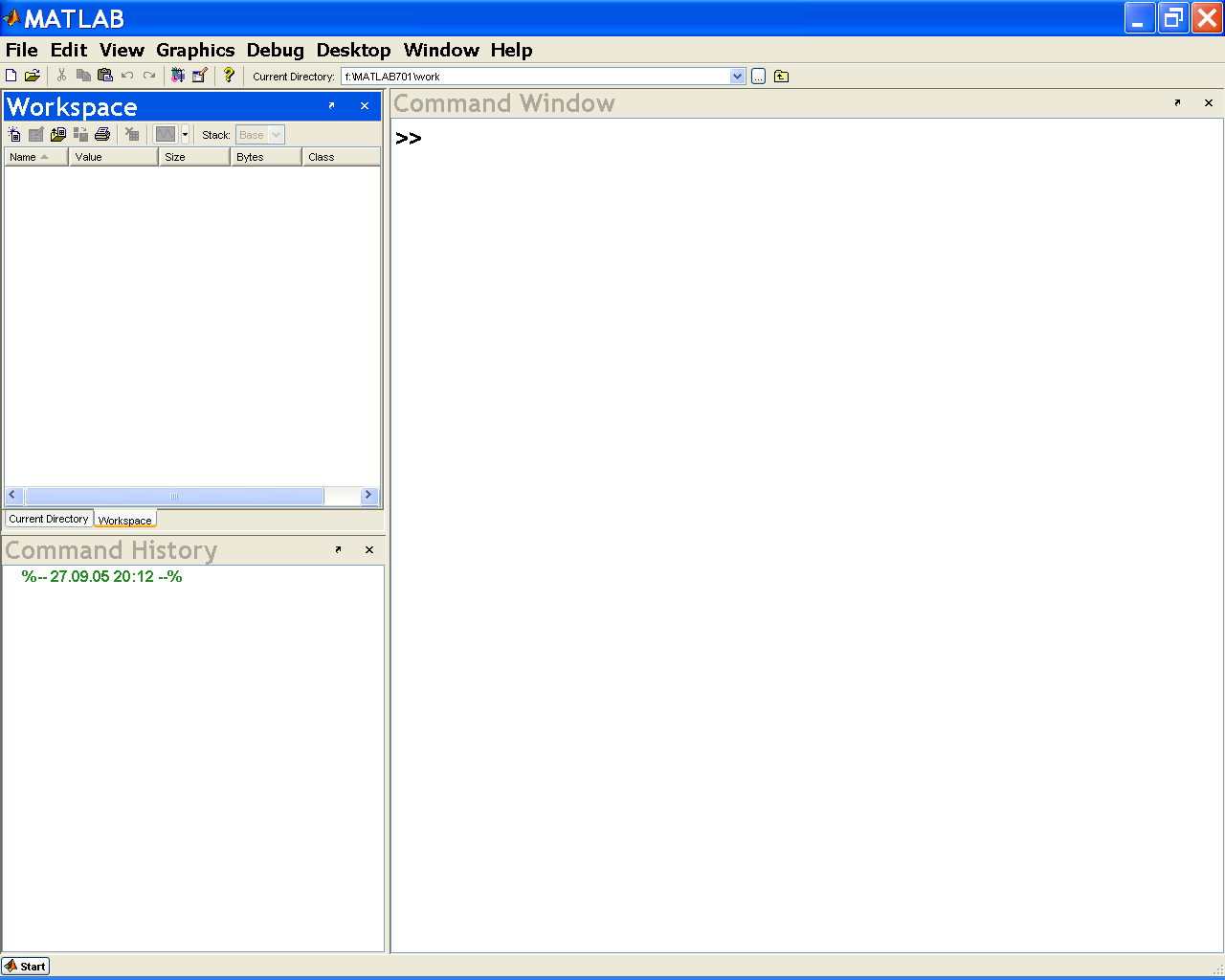


Рис. **3.1**-5. **Основное окно** системы MatLab при активном окне **Workspace**

Команды элементов основного меню **View**и**Graphics**показаны на рис**3.1**-6.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  |  | |

Рис. **3.1-6.** Команды элементов главного меню системы MatLabпри активном

окне **Workspace**

Окно **Workspace(**рис. **3.1**-**6**), предна­значено для быстрого просмотра атрибутов переменных, располагающихся в рабочей области. С помощью окна **Workspace** можно уви­деть имя переменной (**Name**), значение (**Value**), ее размер (**Size**), число байтов (**Bytes**), занимаемых переменной в памяти, и ее класс (**Class**). Для идентификации клас­са переменной слева от имени исполь­зуется соответствующая иконка.

Окно **Workspace** можно отобразить на экране либо с помощью соответствующей команды меню **Desktop,**либо с помощью задания команды **Workspace** в командном окне. С помощью элементов основного меню **View** 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. Более быстрый способ сортировки переменных по ат­рибутам реализуется щелчком левой кнопки мыши на имени соответствую­щего атрибута.

Строка инструментов окна **Workspace** позволяет выполнять следующие команды: создание новой переменной (**Newvariable**); открытие редактора данных с целью просмотра или редактирования значений выделенной переменной (**OpenSelection**); загрузка данных из файла в рабочую область (**LoadDataFile**); сохранение рабочей области в файле (**Save**); печать содержимого рабочей области (**Print**); удалять выделенную переменную (**Delete**); построение различных типов графиков (**plot(d)**); переключение с помощью списка **Stack** между основной рабочей обла­стью (**Base)** и рабочей областью функций во время их отладки (**Stack**).

Просматривать содержимое рабочей области, загружать и удалять дан­ные можно не только с помощью средств окна **Workspace,** но также с помо­щью **CommandWindow.** Для просмотра имен переменных, находящихся в рабочей области, необходимо в командную строку ввести команду **who**. Результат задания команды **who** соответствует внешнему виду окна **Work­space**при отключенных полях **Size,Bytes** и **Class**. Полную информа­цию о содержании рабочей области можно получить с помощью команды: **whos**. В результате информация, выведенная в командное окно, будет соот­ветствовать представленной в окне просмотра рабочей области при всех включенных полях, с тем исключением, что в командное окно будет выведе­но общее количество переменных и общий объем занимаемой ими памяти. Для удаления переменной из рабочей области необходимо ввести в командную строку команду

|  |
| --- |
| **Пример3.1-50** |
| **>>clearимя\_переменной**  **>>** |

Очистка рабочей области осуществ­ляется с помощью команды **clear** без параметров. Загрузка всех данных из файла реализуется командой **load***имя\_файла*, а выборочная загрузка - ко­мандой **load***имя\_файлаимя\_\_переменной***.**

Для сохранения рабочей области на диске необходимо ввести команду **save***имя\_файла***.**

Данные будут сохране­ны в файле с расширением **.mat**. Выборочное сохранение переменных из рабо­чей области обеспечивается командой **save***имя\_файлаимя\_переменной***.**

Редактор данных, изображенный на рис **3.1**-4, предназначен для про­смотра и редактирования значений переменных. Под редактированием переменных подразумевается не только изменение значений элементов массива, но также и изменение размера массива.

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

Окно истории команд, показанное на рис**.3.1**-1 служит для просмотра команд, заданных ранее в командной строке **CommandWindow.** В окне исто­рии команд можно также просмотреть дату и время начала сеанса работы с системой MatLab. Сеанс работы с системой MatLab начинается после ее загрузки в память и вывода на экран ее рабочего стола. Завершение сеанса работы сопровождается закрытием основного окна системы MatLab.

С помощью контекстного меню окна истории команд, изображенного на рис. **3.1**-7 можно выполнять следующие действия: вырезать (**Cut**) и копировать (**Сору**) выделенные строки в буфер обмена; повторно выполнять команду или серию выделенных команд (**EvaluateSelection**); создавать новый **m**-файл (**CreateM-File**) и **Shortcut** путем копирования выделенных строк; а также удалять выделенные строки (**DeleteSelection**); удалять все строки из окна истории команд до выделенной строки (**DeletetoSelection**) и полностью очищать окно истории команд (**DeleteEntireHistory**).



Рис. **3.1**-7. Контекстное меню окна истории команд

Изменять параметры работы окна истории команд можно с помощью диалогового окна **Preferences**, представленного на рис. **3.1**-3, при активном инструментальном средстве **CommandHistory**. Диалоговое окно вызывается из меню **File** командой **Preferences...** В окне присутствуют две группы опций: **Settings**(Настройки) и**Saving**(Сохранение). В первой группе опция включа­ется установкой флажка напротив ее имени. Во второй группе опции пере­ключаются при щелчке на соответствующем имени.

В группе **Settings** можно включать или выключать следующие опции: **Saveexit/quitcommand**– сохра­нять в истории команду**exit/quit**(при задании одной из этих команд система **MatLab** завершает работу);**Saveconsecutiveduplicatecommands**– сохранять одинаковые команды, заданные в командном окне друг за дру­гом; **Savecommandsenteredataninputprompt**– сохранять команды, введенные в командную строку в ответ на команду **input**.

|  |
| --- |
| **Пример3.1-51** |
| **>> n=input (' Введите номер варианта')};**  **Allow Drag and Drop editing**  **>>** |

В группе **Saving** можно переключаться между следующими опциями: **Savehistoryfileonquit**  – сохранять файл истории при выходе (история команд сохраняется в файле **history.m**); **Saveafter n commands**  – сохранять файл истории после ввода каждой n команды; **Don'tsavehistoryfile** – не сохранять файл истории команд (несмотря на включенную опцию, в течение сеанса работы в окне истории команд сохраняются все команды, которые будут удалены при выходе).

Окно **CommandHistory** хранит все команды, набираемые пользователем. В отличие от содержимого **CommandWindow**сюда не попадают сообщения системы и результаты вычислений.

##### Простейшие средства программированиям в MatLab

До сих пор мы в основном использовали систему **MatLab** в режиме непосредственного счета – *в* **командном режиме (командном окне)***.* Однако при решении серьезных задач возникает необходимость сохранения используемых последовательностей вычислений, а также их дальнейшей модификации. Иными словами, существует необходимость **программирования** решения задач. Программирование в системе MatLab является эффективным средством ее расширения и адаптации к решению специфических проблем. Оно реализуется с помощью **языка программирования** системы MatLab.

Большинство объектов этого языка, в частности все **команды, операторы** и **функции**, одновременно являются объектами **входного языка общения с системой в командном режиме работы**. Так что фактически мы приступили к описанию языка программирования системы MatLab с первых строк данной книги.

Так в чем же отличие входного языка от языка программирования? В основном – в способе фиксации создаваемых ими кодов. Сессии в командном режиме работы не сохраняются в памяти компьютера. А вот программы на языке программирования MatLab сохраняются в виде текстовых файлов (**m**-**файлов)**. При этом могут сохраняться как целые программы в виде **файлов-программ**, так и отдельные **программные модули — функции**. Кроме того, важно, что программа может менять структуру алгоритмов вычислений в зависимости от входных данных и данных, создаваемых в ходе вычислений.

Итак, программами в системе MatLab являются программы оформленные как **m**-**файлы** текстового формата, содержащие запись программ в виде программных кодов. Язык программирования системы MatLab имеет следующие средства:

* данные различного типа;
* константы и переменные;
* операторы, включая операторы математических выражений;
* встроенные команды и функции;
* функции пользователя;
* управляющие структуры;
* системные операторы и функции;
* средства расширения языка.

Тексты программ в системе MatLab пишутся на языке высокого уровня, достаточно понятном для пользователей умеренной квалификации в области программирования. Язык программирования MatLab является типичным **интерпретатором***.*

Интерпретация означает, что MatLab не создает исполняемых конечных программ. Они существуют лишь в виде **m**-**файлов**, для выполнения, которых необходима среда MatLab. Однако для программ на языке MatLab созданы компиляторы, транслирующие программы MatLab в коды языков программирования С и C++. Это решает задачу создания исполняемых программ, первоначально разрабатываемых в среде MatLab. Компиляторы для системы MatLab являются вполне самостоятельными программными средствами.

Начальное представление о **переменных, встроенных константах и функциях** уже было дано в предшествующей теме.

Итак, мы установили, что работа в командном режиме (сессия) не является программированием. Внешним атрибутом последнего в MatLab служит задание последовательности действий по программе, записанной в виде **m**-**файла**. Для создания **m**-**файлов** может использоваться как редактор, встроенный в MatLab, так и любой текстовый редактор. Подготовленный и записанный на диск **m**-**файл** с соответствующим именем становится частью системы, и его можно вызывать как из командной строки, так и из другого **m**-файла. Имеется два типа **m**-файлов: **файлы-программы** и **файлы-функции**. Важно, что в процессе своего создания они проходят синтаксический контроль с помощью встроенного в систему MatLab **редактора/отладчика m**-**файлов**.

Здесь мы рассмотрим лишь простейший тип m-файлов – файлы-программы, а все средства программирования в MatLab студенты могут изучить самостоятельно (Тема 3.5. М-файлы и программирование средствами MatLab).

**Файл*-*программа***,* именуемый также **Script-файлом**, является просто записью последовательности команд без входных и выходных параметров. Он имеет следующую структуру:

|  |
| --- |
| **-Имя\_файла** |
| **% Основной комментарий**  **% Дополнительный комментарий**  ***Тело файла с любыми выражениями*** |

Файлы-программы имеют следующие особенности:

* они не имеют входных и выходных аргументов;
* работают с данными из рабочей области;
* в процессе выполнения не компилируются;
* представляют собой зафиксированную в виде файла последовательность операций, полностью аналогичную той, что используется во время сессии.

Основным комментарием является первая строка текстовых комментариев, а дополнительным – последующие строки комментариев. Это связано с тем, что основной комментарий выводится при выполнении команд **lookfor** и **helpимя\_каталога**. Полный комментарий выводится при выполнении команды **helpИмя\_файла**. В качестве примера рассмотрим файл-программу **PlotSin**и пример вызова ее.

|  |
| --- |
| **Пример 3.1-52 m-файла** |
| **% Построение графика синусоиды линией красного цвета**  ***% с* масштабной сеткой в интервале [xmin.xmax]**  **x=xmin:0.1:xmax;**  **plot(x,sin(x),е6'r')**  **gridon** |

|  |
| --- |
| **Пример 3.1-52 сессии** |
| **>>xmin=-1;**  **>>xmax=1;**  **>> Пример5.5-1;**  **>>** |

Первые две строки **m**-файла **Пример 3.1-52–** это комментарий, остальные – тело файла. Знак ***%***в комментариях должен начинаться с первой позиции строки. В противном случае команда **help*name\_файла*** не будет воспринимать комментарий и возвратит сообщение вида

**No help comments found in-name.m.**

Обратите внимание на то, что такой файл нельзя запустить без предварительной подготовки, сводящейся к заданию значений переменным **xmin** и **хmах**, которые используются в теле файла. Это следствие первого свойства файлов-программ – они работают с данными из рабочей области. Переменные, используемые в файлах-программах, являются глобальными, т. е. они действуют одинаково в командах сессии и внутри программного блока, которым является файл-сценарий. Поэтому заданные в сессии значения переменных используются и в теле файла. Имена файлов-программ нельзя использовать в качестве параметров функций, поскольку файлы-программы не возвращают значений. Можно сказать, что файл-программа – это простейшая программа на языке программирования MatLab.

Результаты работы сессии Примера **3.1-52**. приведены на рис. 3.1-8.

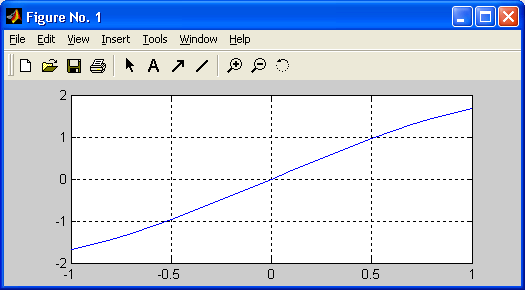


Рис. 3.1-8. Результат работы файла-программы Пример **3.1-52**