Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Южно-Уральский государственный университет
Кафедра «Строительная механика»

681.3.06(07)
П64

А.Н. Потапов, Е.М. Уфимцев

МАТЕМАТИЧЕСКАЯ СИСТЕМА MATLAB

Учебное пособие для самостоятельной работы

Часть первая

Челябинск
Издательство ЮУрГУ
2009

УДК [681.3.066:624.04](075.8)
П64

Одобрено
учебно-методической комиссией
архитектурно-строительного факультета

Рецензенты:
С.Б. Шматков, А.Ю. Рыжков

Потапов, А.Н.

П64 Математическая система MATLAB: учебное пособие для самостоятельной работы / А.Н. Потапов, Е.М. Уфимцев. – Челябинск: Изд-во ЮУрГУ, 2009. – Ч. 1. – 75 с.

Изложены приемы работы в диалоговом режиме с системой MATLAB седьмой версии. Дано описание работы с матрицами и массивами. Приведен синтаксис основных команд, функций и операторов системы. Учебное пособие содержит большое количество примеров с подробными комментариями.

Ориентировано на самостоятельную работу студентов архитектурно-строительного факультета специальности 270102 – «Промышленное и гражданское строительство» специализации «Исследование и проектирование зданий и сооружений». Может быть использовано на других архитектурно-строительных специальностях, а также аспирантами и научными работниками.

УДК [681.3.066:624.04](075.8)

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Настоящее учебное пособие предназначено для студентов АС факультета специальности 270102 – «Промышленное и гражданское строительство» специализации «Исследование и проектирование зданий и сооружений». Оно ориентировано на самостоятельное изучение основных элементов многофункциональной интегрированной системы автоматизации математических и научно-технических расчетов MATLAB версии 7.2 при выполнении как исследовательских работ, так курсовых и дипломных проектов. При этом следует учесть, что круг вопросов, затронутых в данном пособии, ограничен рассмотрением работы системы MATLAB в диалоговом режиме (в режиме калькулятора).

Пособие содержит пять разделов по описанию приемов работы с системой MATLAB. В первом разделе представлено описание интерфейса программы, основных ее команд и функций. Второй раздел посвящен элементарным сведениям о действиях с действительными и комплексными числами в стандарте IEEE-арифметики; здесь содержится информация о константах и переменных, об основных операторах, командах и функциях.

В последующих разделах, с третьего по пятый, сосредоточены сведения об особенностях работы с более сложными математическими объектами – векторами и матрицами. В третьем разделе изложены способы задания векторов и матриц, основные приемы их формирования и ряд операций, связанных с преобразованием матриц (расширением, уменьшением, вставкой, удалением и т.д.). Здесь же рассмотрены функции формирования векторов и матриц специального вида и основные операции с матрицами и массивами. В четвертом разделе приведены основные функции для обработки данных, представленных массивами. И, наконец, пятый раздел посвящен наиболее часто используемым функциям линейной алгебры при решении самых разнообразных задач инженерной практики.

Необходимость в создании пособия продиктована, главным образом, тем, что обширная справочная литература MATLAB пока еще остается труднодоступной для неанглоязычных пользователей. Кроме того, авторы постарались дать более строгую систематизацию материала, касающегося работы с массивами и матрицами.

В пособии приведен исчерпывающий синтаксис основных команд, функций и операторов системы. Для максимальной доступности излагаемого материала текст снабжен большим количеством примеров, сопровождающихся необходимыми комментариями.

ВВЕДЕНИЕ

Система MATLAB (**MAT**rix **LAB**oratory – **МАТ**ричная **ЛАБ**оратория) разработана фирмой The MathWorks, Inc. (США, г. Нейтик, шт. Массачусетс) около тридцати лет назад. В настоящее время данная система получила широкое распространение во всем мире и является мощным и универсальным средством решения различных инженерных и научных задач. Спектр проблем, изучение которых может быть реализовано на основе пакета MATLAB, охватывает матричный анализ, обработку сигналов и изображений, задачи математической физики, оптимизационные задачи, обработку и визуализацию данных и многие другие.

Историю появления пакета MATLAB связывают с именем профессора Клива Б. Моулера (Cleve B. Mouler). До перехода в фирму MathWorks он занимался преподавательской и научно-исследовательской деятельностью на кафедрах математики и в компьютерных центах ряда университетов США (Нью-Мехико, Мичиган, Стэнфорд). Около 30 лет назад Моулер принимал участие в разработке пакетов программ на языке Fortran для решения задач линейной алгебры (LINPACK) и исследованиях проблемы собственных значений матриц (EISPACK). В 1980 г. на международной конференции AFIPS он представил доклад "Design of an interactive matrix calculator", в котором, по-видимому, впервые было озвучено название MATLAB.

Второе рождение пакета MATLAB связывают с Джеком Литтлом (Jack Little) - нынешним президентом фирмы MathWorks, который в начале 80-х годов прошлого века перенес программу MATLAB на более современные вычислительные платформы VAX, Macintosh и IBM PC. Дальнейшее развитие пакета происходило под эгидой MathWorks, однако к расширению состава пакета и сфер его применения были привлечены коллективы высококвалифицированных математиков и инженерно-технических работников Старого и Нового света.

С момента основания фирмы (1984 г.) К. Моулер является ее бессменным научным руководителем. Модернизацию и программное сопровождение пакета MATLAB обеспечивают более 1000 сотрудников фирмы MathWorks.

К концу 2008 г. реализации пакета MATLAB насчитывают следующие версии (начиная с 5-й):

1. MATLAB 5.0 – декабрь 1996 г.
2. MATLAB 5.1 – май 1997 г.
3. MATLAB 5.3 – январь 1999 г.
4. MATLAB 6.0 – ноябрь 2000 г.
5. MATLAB 6.1 – июнь 2001 г.
6. MATLAB 6.5 – июнь 2002 г.
7. MATLAB 6.51 – август 2003 г.
8. MATLAB 7.0 – июнь 2004 г.
9. MATLAB 7.0.1 – сентябрь 2004 г.
10. MATLAB 7.0.4 – март 2005 г.
11. MATLAB 7.1 – сентябрь 2005 г.
12. MATLAB 7.2 – март 2006 г.
13. MATLAB 7.3 – сентябрь 2006 г.
14. MATLAB 7.4 – март 2007 г.
15. MATLAB 7.5 – сентябрь 2007 г.
16. MATLAB 7.6 – март 2008 г.
17. MATLAB 7.7 – октябрь 2008 г.

Одной из наиболее важных особенностей пакета MATLAB является его открытость. Комплект поставки содержит довольно много исходных текстов программных модулей, функций, тестовых примеров. Это предоставляет возможность пользователям разобраться в алгоритмах, модифицировать их для своих приложений и расширять сферу применения пакета.

Основной объект MATLAB – прямоугольный числовой массив, допускающий комплексные элементы и ввод матриц, не требующий явного указания их размеров. При этом любая переменная по умолчанию воспринимается системой как вектор или матрица. Даже обычные числа и переменные (скаляры) в MATLAB рассматриваются как одноэлементные матрицы (порядка 1×1), что дает единообразные формы и методы проведения операций над обычными числами и массивами.

В MATLAB реализованы классические численные алгоритмы решения уравнений, задач линейной алгебры, решения нелинейных уравнений и задач оптимизации, нахождения значений определенных интегралов, интерполяции, решения обыкновенных дифференциальных уравнений (ОДУ) и дифференциальных уравнений в частных производных и другие алгоритмы.

Работа в среде MATLAB может осуществляться в двух режимах. В режиме непосредственных вычислений без какого-либо программирования. Это так называемый режим интерпретации команд и операторов, когда последние вводятся в ходе сеанса в командной строке, а MATLAB выполняет их немедленную обработку и выдает вычисленный результат. Другой режим работы состоит в возможности обработки последовательности команд и операторов на языке MATLAB в виде подготовленного m-файла. При вызове соответствующего m-файла обеспечивается ввод данных, организация вычислений и вывод результатов на экран, в результате чего реализуется программный режим.

В обоих режимах реализуются практически все вычислительные возможности системы, в том числе по выводу информации в графической форме. Программный режим позволяет сохранять разработанные вычислительные алгоритмы и, таким образом, повторять вычисления при необходимости.

Система MATLAB имеет собственный язык программирования, напоминающий языки Basic и C. Запись программ в системе является традиционной и поэтому привычна для большинства пользователей персональных компьютеров. К тому же система дает возможность редактировать программы при помощи любого привычного для пользователя текстового редактора.

Работа с такой мощной математической системой, как MATLAB, требует от пользователя соответствующей теоретической подготовки, без чего невозможно правильное применение используемых в системе методов и корректность получаемых результатов. Применение базовых вычислительных возможностей требуют знаний математического аппарата и основных численных методов в рамках программы технических специальностей вузов. К тому же, сведения, изложенные в справочной системе, оказываются ценной информацией для желающих самостоятельно разобраться в обширных возможностях пакета MATLAB.

1. ИНТЕРФЕЙС MATLAB И КОМАНДЫ ОБЩЕГО НАЗНАЧЕНИЯ

1.1. Основные элементы рабочей среды

При первом запуске MATLAB на экране открывается рабочая среда, изображенная на рис. 1. Основными элементами ее (рис. 2) являются:

- выпадающие меню;
- стандартная панель инструментов с кнопками и раскрывающимся списком (**Desktop Toolbar**);
- пользовательская панель инструментов (**Shortcuts Toolbar**);
- окно команд (**Command Window**);
- окно рабочего пространства переменных (**Workspace**);
- окно истории команд (**Command History**);
- окно выбора текущего рабочего каталога (**Current Directory**);
- строка состояния с кнопкой **Start**.

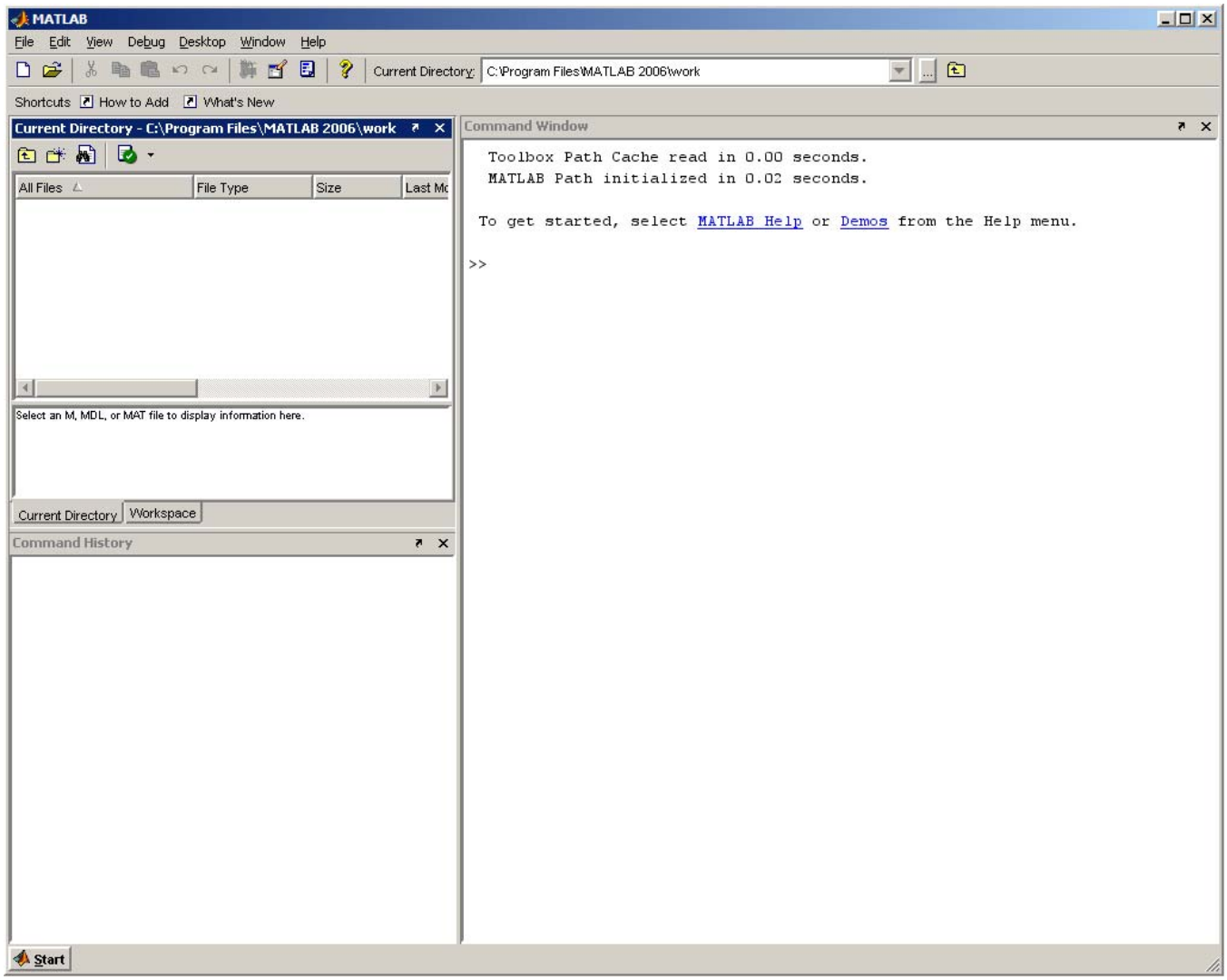

Рис. 1. Рабочая среда MATLAB при первом запуске

1.2. Инструментальные окна

Наиболее часто используемым является окно **Command Window** (рис. 2). Оно предназначено для ввода пользовательских команд с последующим их выполнением, а также для отображения результатов выполнения этих команд.

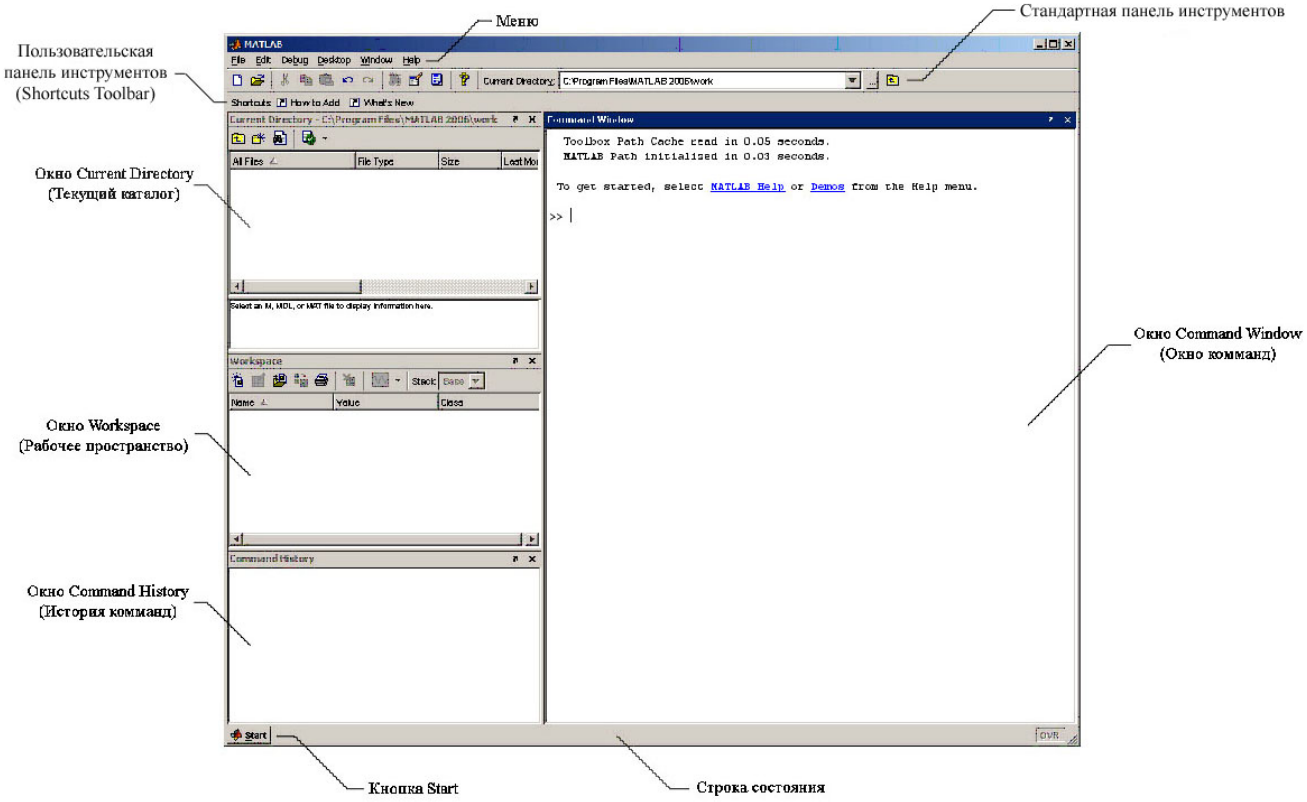

Рис. 2. Основные элементы рабочей среды MATLAB

Окно **Workspace** (рис. 3) отображает текущий набор переменных, созданных пользователем. Это окно предоставляет следующие данные о переменных рабочего пространства (колонки):

- **Name** – имя переменной;
- **Value** – значение переменной;
- **Size** – размер переменной (для скалярного значения – 1×1, для массива – m×n);
- **Bytes** – занимаемый переменной объем памяти в Байтах;
- **Class** – тип переменной (integer, single, double, cell и т. д.).

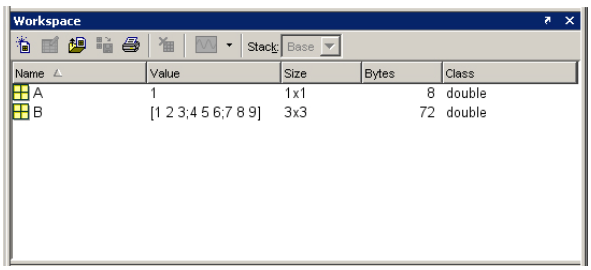

Рис. 3. Окно **Workspace**

Также в этом окне можно откорректировать имя переменной (**Name**) или ее значение (**Value**). Причем, при корректировке значения вызывается специальное окно редактора массивов **Array Editor** (рис. 4). Это новый инструмент, появившийся в 7-й версии и внешне похожий на электронную таблицу Microsoft Excel.

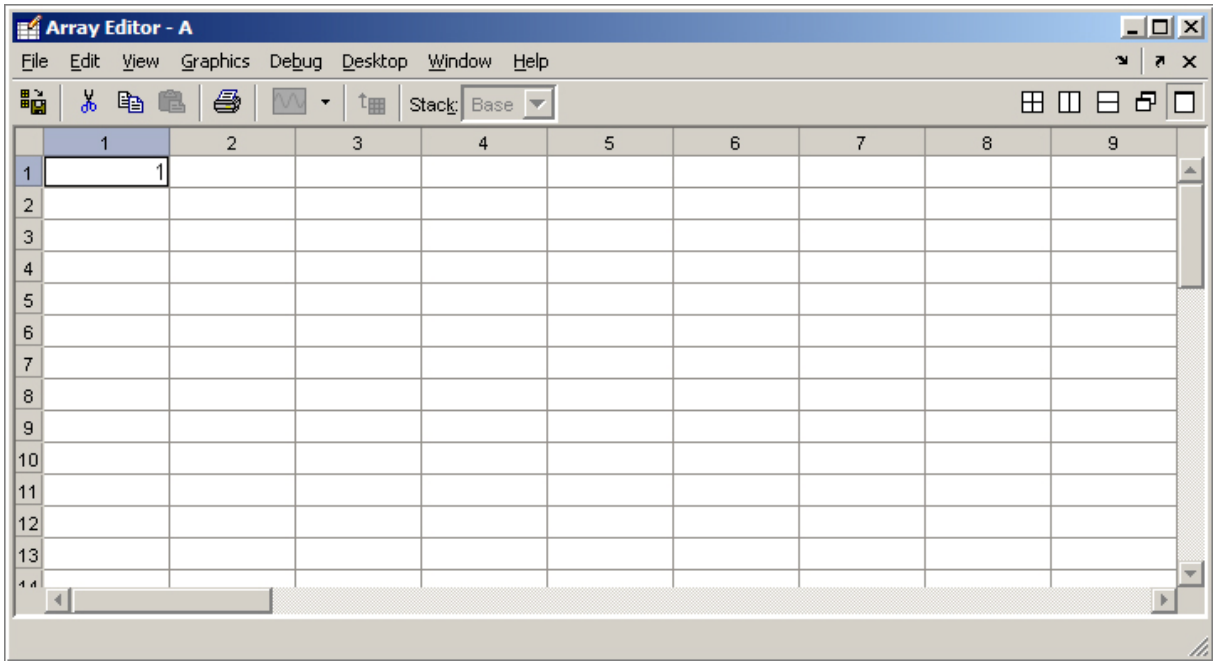

Рис. 4. Окно **Array Editor**

Окно **Command History** хранит все команды, вводимые пользователем в окне **Command Window**, однако в отличие от содержимого окна **Command Window**, сюда не попадают сообщения системы и результаты вычислений. Эта информация может оказаться полезной для формирования программы, исполняемой в автоматическом режиме.

Окно **Current Directory** предназначено для выбора текущего рабочего каталога и отображения его содержимого – подкаталогов и файлов. Кроме этого, есть возможность просматривать заголовочные комментарии m- и mdl-файлов и содержимое mat-файлов.

В строке состояния, как понятно из названия, отображается информация о текущем состоянии среды. При нажатии на кнопку **Start** открывается меню, приведенное на рис. 5. С его помощью обеспечивается доступ ко всем основным средствам MATLAB.

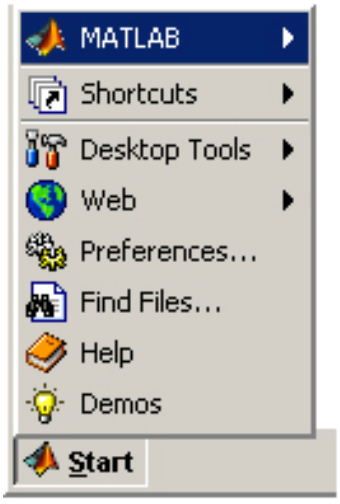

Рис. 5. Кнопка и меню **Start**

1. 3. Главное меню

Далее рассмотрим команды главного меню (рис. 6).

Рис. 6. Главное меню

1.3.1. Меню File. Команды меню **File** (Файл) (рис. 7) выполняют стандартные для большинства систем программирования функции. Первая группа команд позволяет создать новый файл (**New**) или открыть уже существующий (**Open**). Команда **Close Command Window** позволяет закрыть окно команд.

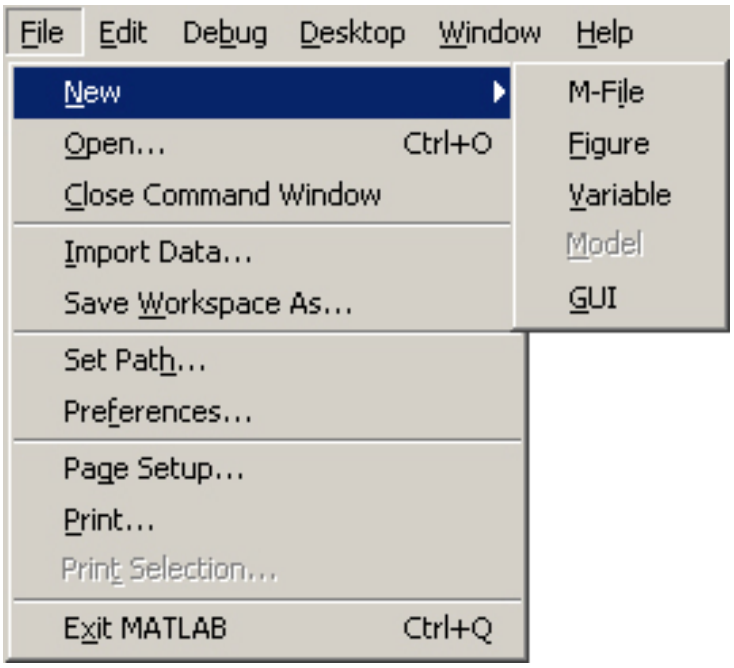

Рис. 7. Команды меню **File**

Команды второй группы позволяют импортировать ранее сохраненные данные в рабочее пространство (**Import Data**) или сохранить значения всех переменных рабочего пространство в дисковом файле (**Save Workspace As**).

Команда **Set Path** позволяет пополнить список каталогов, просматриваемых системой, или изменить порядок их просмотра. Настройка параметров системы выполняется в окне предпочтений **Preferences** (рис. 8), вызываемом соответствующей командой меню. Подробнее о настройках системы рассказано в разделе «Настройка среды MATLAB».

Четвертая группа команд – традиционная для большинства систем. Она обеспечивает получение твердой копии данных (**Print** или **Print Selection**) с предварительной настройкой параметров бумаги и принтера (**Page Setup**).

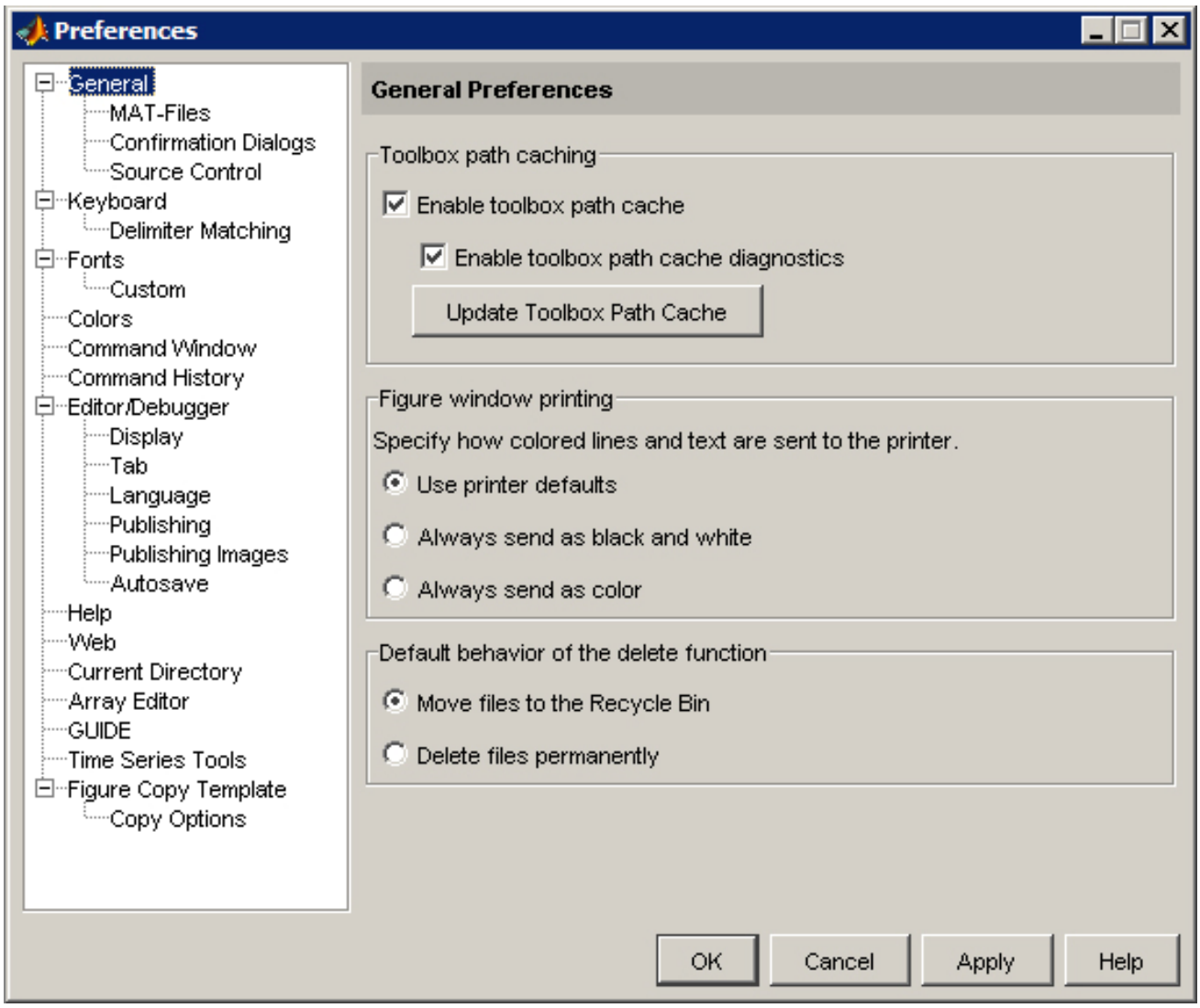

Рис. 8. Окно **Preferences**

1.3.2. Меню Edit. Пункт меню **Edit** (Правка) (рис. 9) содержит команды для работы с текстовыми данными (большинство команд данного меню являются характерными для любого текстового редактора):

- вырезать (**Cut**);
- копировать (**Copy**);
- вставить (**Paste**);
- вставить в рабочее пространство (**Paste to Workspace**);
- выделить все (**Select All**);
- удалить (**Delete**).

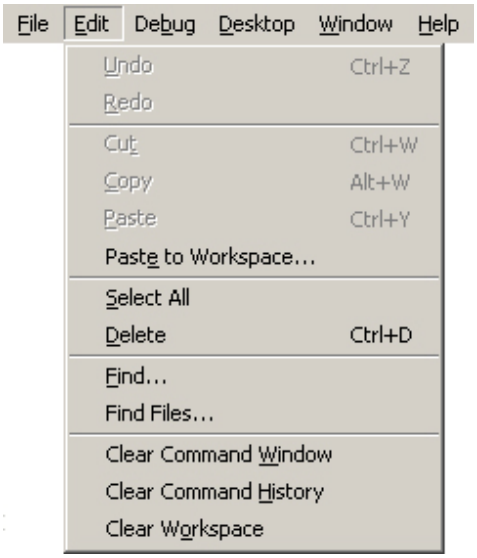

Рис. 9. Команды меню **Edit**

Команды **Undo** (Отменить) и **Redo** (Повторить) используются, соответственно, для отмены только что совершенного действия или отказа от предшествующей отмены.

С помощью команды **Find** (Найти) можно найти текстовый фрагмент в файле и, при необходимости, произвести его замену. Команда **Find Files** (поиск файлов) позволяет выполнить поиск файлов по заданным параметрам.

Последняя группа команд **Clear Command Window**, **Clear Command History** и **Clear Workspace** позволяет произвести очистку содержимого соответствующих окон – **Command Window**, **Command History** и **Workspace**.

1.3.3. Меню Debug. Меню **Debug** (Отладка) включает команды, управляющие режимом отладки (рис. 10).

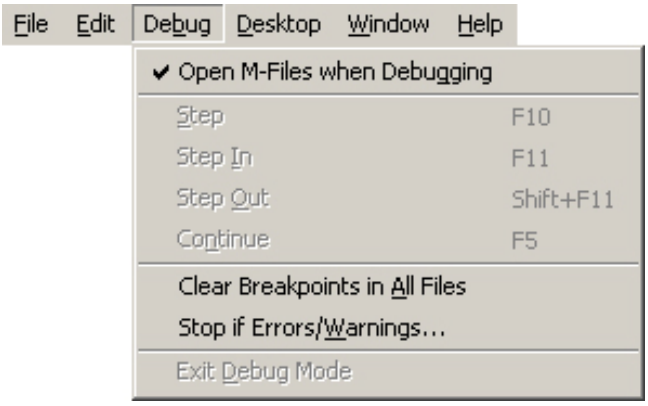

Рис. 10. Команды меню **Debug**

1.3.4. Меню Desktop. Далее рассмотрим команды меню **Desktop** (Рабочий стол) (рис. 11).

Первая группа команд позволяет настроить размер и расположение активного окна рабочей среды:

- команда **Undock …** (Отстыковать …) разъединяет состыкованные окна и позволяет активному окну перемещаться самостоятельно;
- команда **Move …** позволяет пользователю выбрать удобное расположение активного окна;
- команда **Resize …** позволяет пользователю изменить размер активного окна.

Вторая группа позволяет организовать компоновку окон на рабочем столе. Команда **Desktop Layout** (Компоновка рабочего стола) определяет количество и расположение одновременно видимых элементов среды. Вы можете выбрать конфигурацию среды по умолчанию (**Default**), изображенную на рис. 1, либо разместить на экране только окно команд (**Command Window Only**). Команда **History and Command Window** (Окна истории и команд) позволяет сохранить два окна – окно истории команд и окно команд. Команда **All Tabbed** (С вкладками) располагает на экране все окна во вкладках (рис. 12). Команды **User Layout 1**, **User Layout 2** и т. д. появляются в том случае, если пользователь определил соб-

ственную компоновку окон на рабочем столе (см. далее). При этом имена компоновок окон определяются самим пользователем.

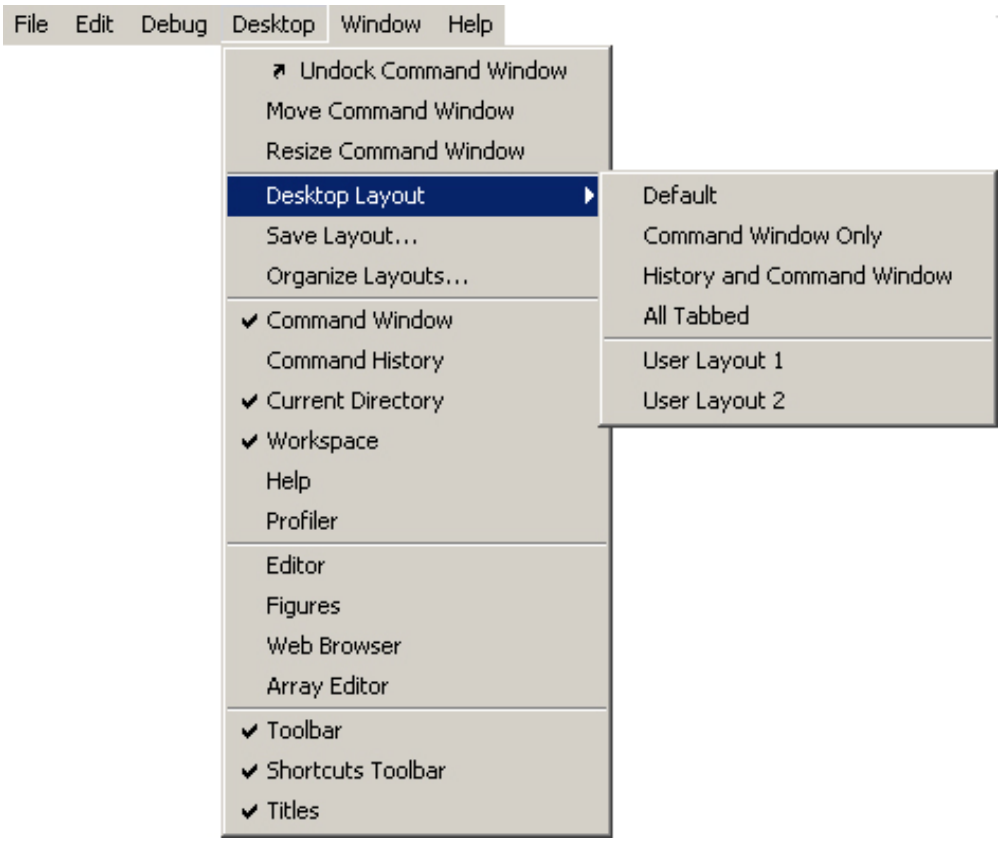

Рис. 11. Команды меню **Desktop**

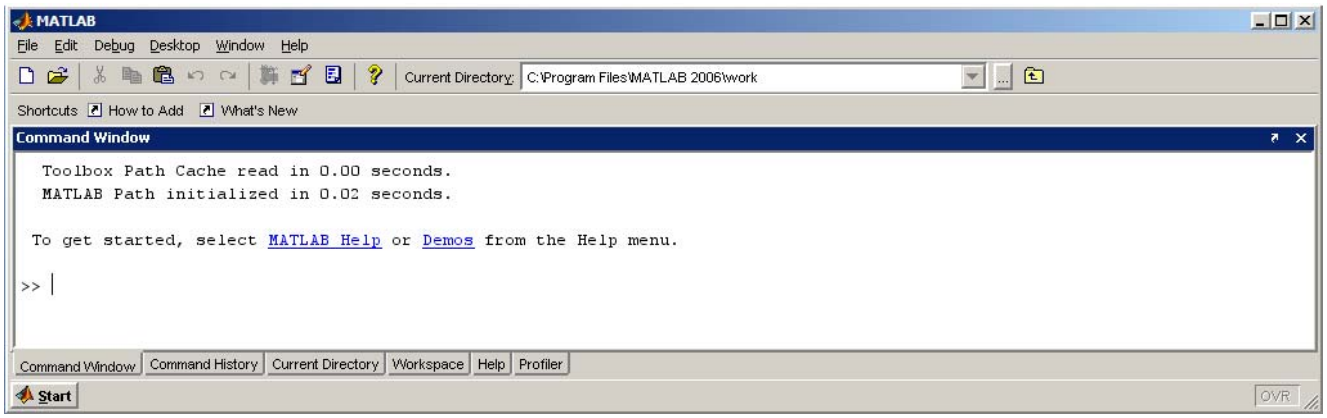

Рис. 12. Конфигурация всех окон во вкладках

При этом одно из окон находится на переднем плане, а любое из оставшихся выходит на передний план после щелчка по соответствующей вкладке.

В MATLAB 7 появилась возможность сохранить ту или иную конфигурацию окон на экране, воспользовавшись командой **Save Layout** (Сохранить компоновку). Для выбора нужной конфигурации необходимо прибегнуть к команде **Organize Layouts** (Организовать разметку) (рис. 13).

Следующие две группы команд позволяют из меню установить комбинацию окон, которые должны присутствовать на экране. Первая группа команд:

- **Command Window;**
- **Command History;**
- **Current Directory;**
- **Workspace;**
- **Help;**
- **Profiler.**

Вторая группа команд:

- **Editor;**
- **Figures;**
- **Web Browser;**
- **Array Editor.**

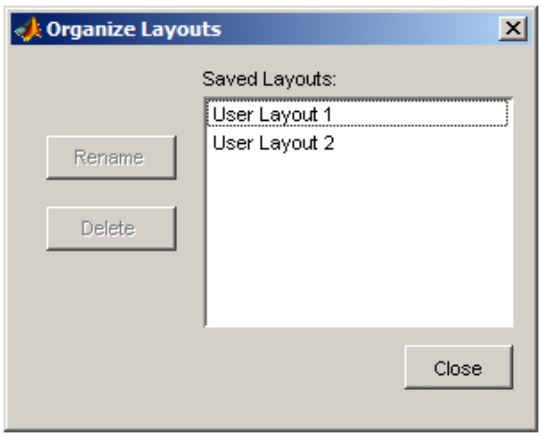

Рис. 13. Окно **Organize Layouts**

Включение или отмена галочки у соответствующей строки меню позволяет управлять появлением или исчезновением того или иного окна.

Команды последней группы управляют видимостью стандартной панели инструментов (**Toolbar**), пользовательской панелью инструментов (**Shortcuts Toolbar**) и заголовков окон (**Titles**) (рис. 14).

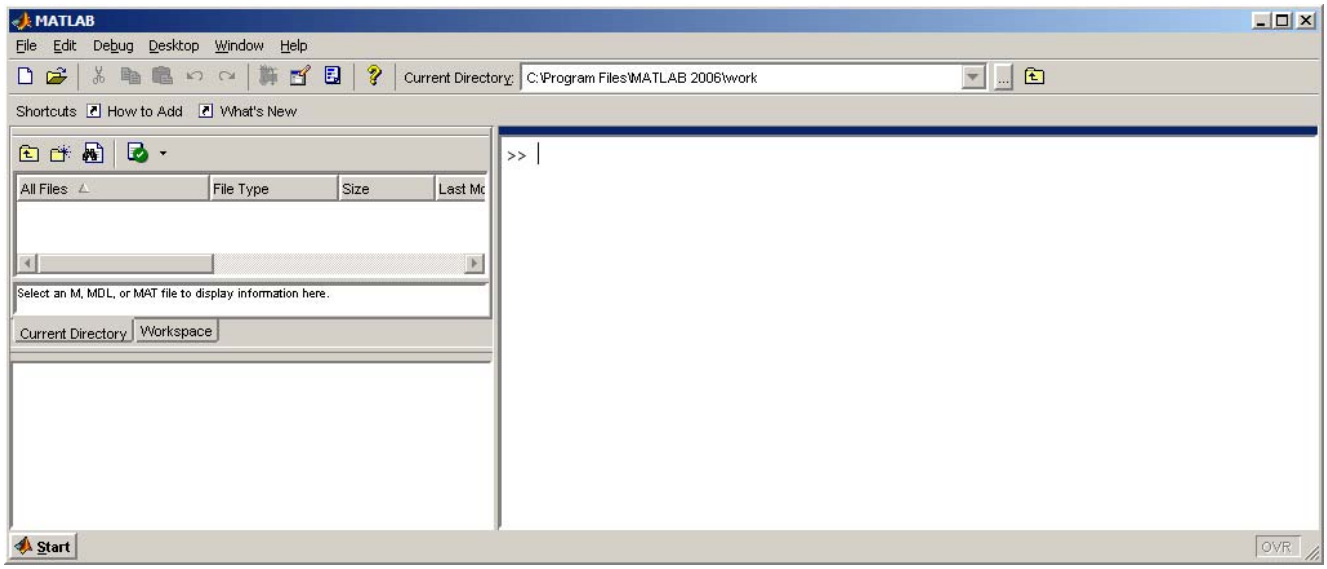

Рис. 14. Рабочая среда MATLAB с отключенными заголовками окон (**Titles**)

1.3.5. Меню Window. С помощью команд меню **Window** (Окно) (рис. 15) можно получить быстрый доступ к открытым окнам среды и документам. Команда **Close All Documents** закрывает все открытые документы (m-файлы, окна **Figure**), кроме основного окна MATLAB.

Следующая группа команд представляет собой список окон, расположенных на рабочем столе (**Desktop**), и таким образом выбор какого-либо элемента этого списка позволяет сделать активным соответствующее окно.

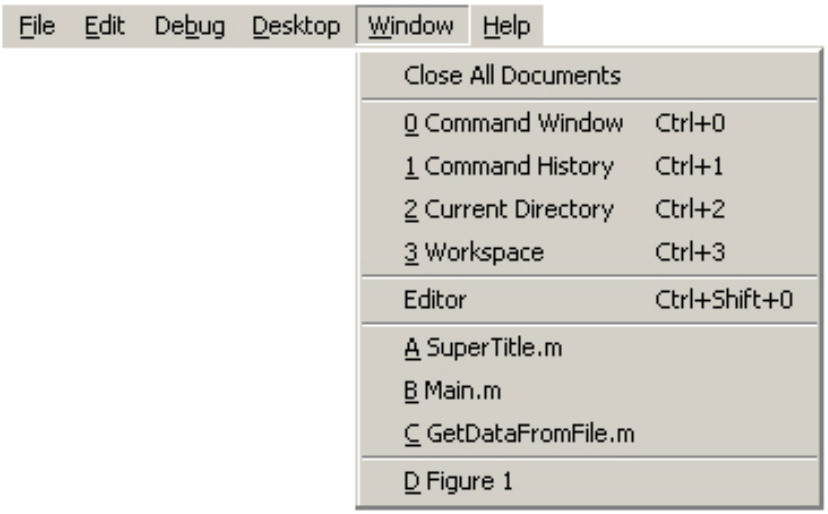

Рис. 15. Команды меню **Window**

Следующие две группы команд доступны только в случае, если открыт один или несколько m-файлов в режиме редактирования. В первой из этих групп команд доступна только команда **Editor**, которая выводит на передний план окно редактора m-файлов **Editor** (рис. 16), о котором будет рассказано в разделе «Редактор исходных кодов». Вторая группа команд представляет собой список, составленный из имен открытых m-файлов. Выбор какого-либо элемента из этого списка выводит на передний план редактор **Editor** с выбранным m-файлом в качестве активного файла для редактирования.

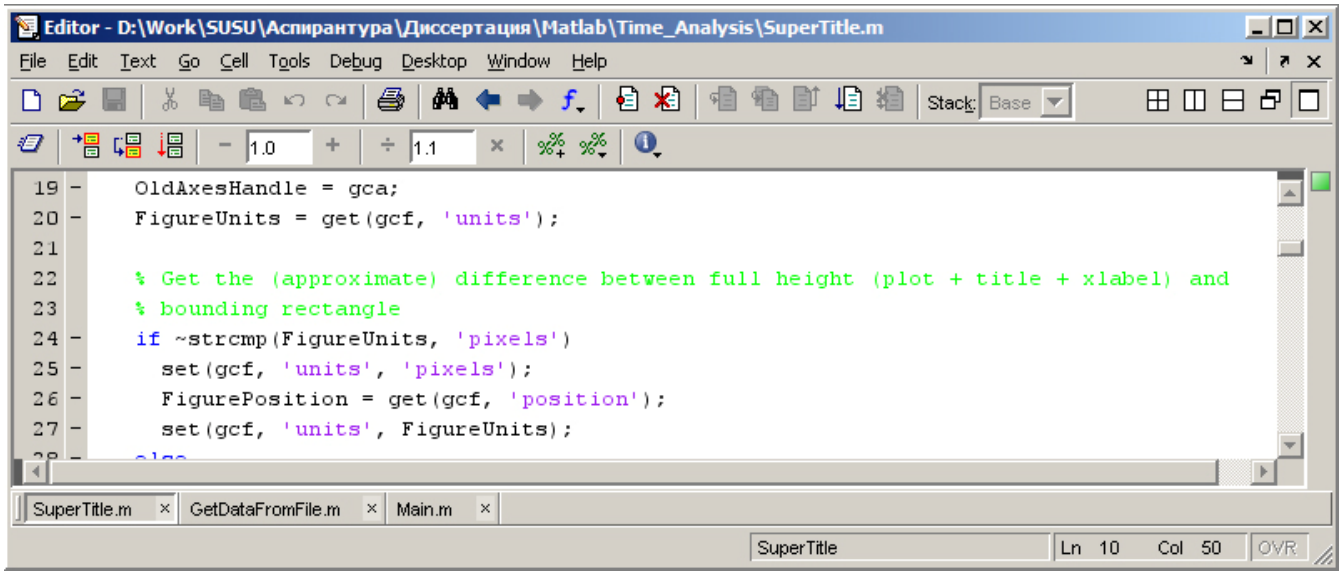

Рис. 16. Окно редактора m-файлов **Editor**

Следующая группа команд доступна только в случае, если открыто одно или несколько графических окон **Figure** (рис. 17) и представляет собой список, составленный из имен открытых графических окон. Выбор какого-либо элемента из этого списка выводит на передний план графическое окно с соответствующим именем.

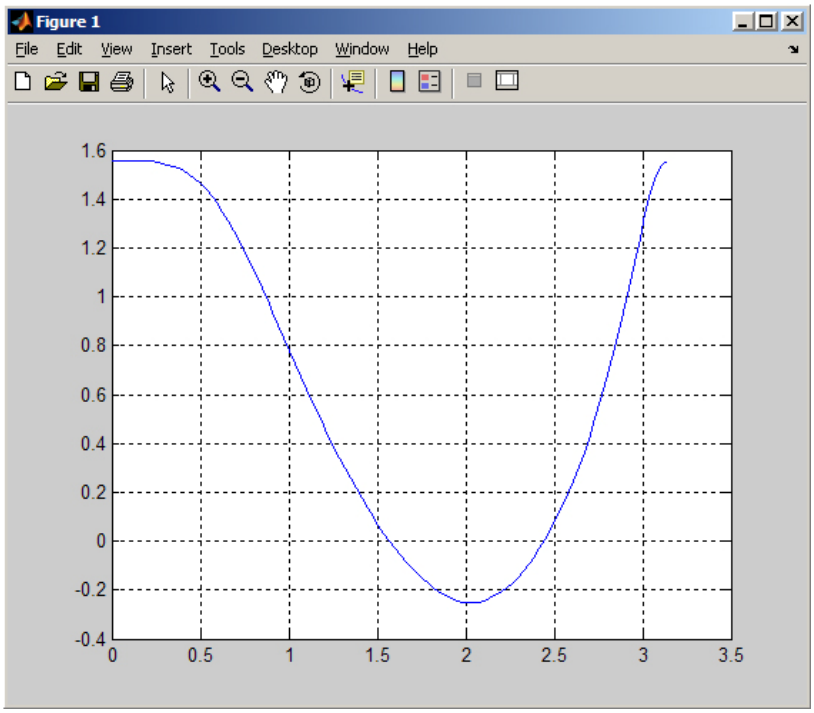

Рис. 17. Графическое окно **Figure**

1.3.6. Меню Help. Меню **Help** (Помощь) (рис. 18) является традиционным для большинства приложений. Команды **Full Product Family Help**, **MATLAB Help**, **Using the Desktop** и **Using the Command Window** позволяют получить доступ к справочной системе MATLAB. Команда **Web Resources** открывает доступ к сетевым ресурсам компании The MathWorks. Команда **Check for Updates** проверяет наличие обновлений в сети. Команда **Demos** открывает окно, знакомящее пользователя с демонстрационными возможностями системы MATLAB. Команды **Terms of Use** и **Patents** открывают окна лицензионного соглашения и патентов, соответственно. Команда **About MATLAB** выводит на экран окно, изображенное на рис. 19.

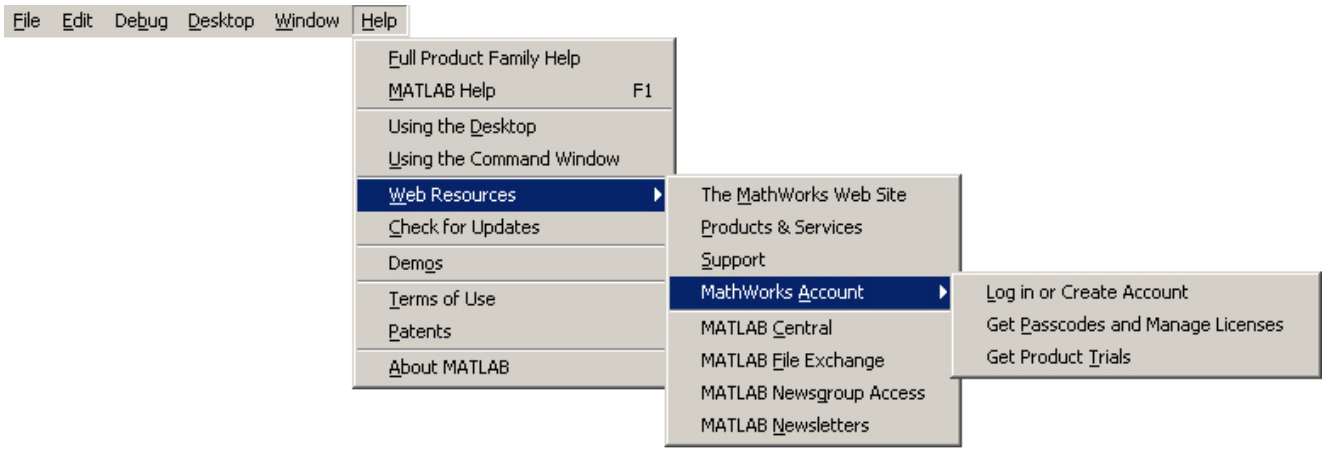

Рис. 18. Команды меню **Help**

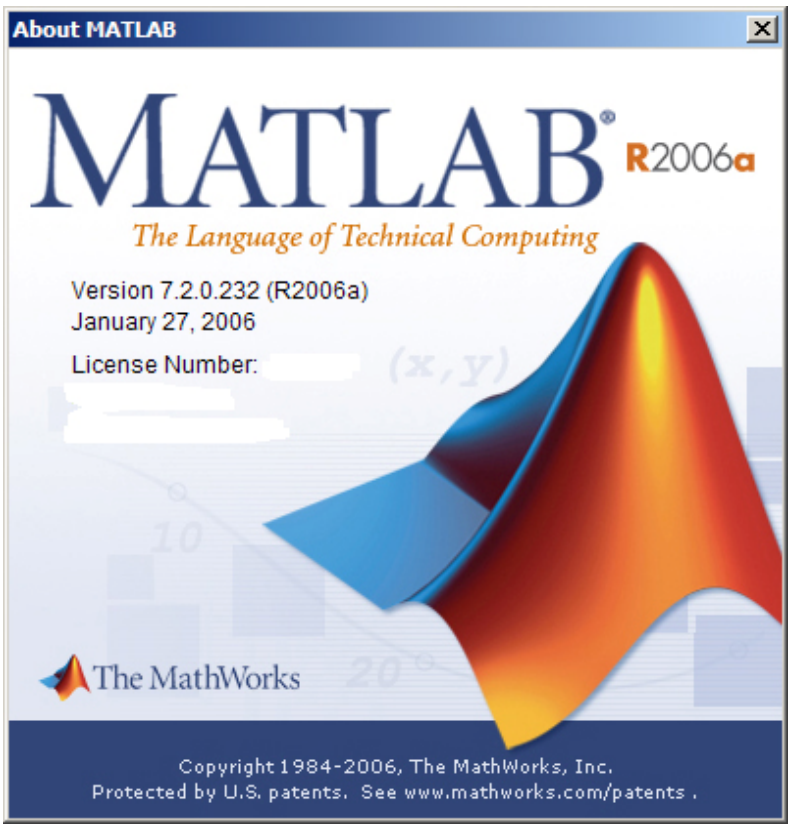

Рис. 19. Окно **About MATLAB**

1.4. Панели инструментов

Далее рассмотрим панели инструментов **Toolbar**:

- стандартная панель инструментов;
- пользовательская панель инструментов **Shortcuts Toolbar**;
- панель инструментов **Current Directory**;
- панель инструментов **Workspace**.

1.4.1. Стандартная панель инструментов. На стандартную панель инструментов (**Desktop Toolbar**) вынесены наиболее употребительные команды главного меню (рис. 20).

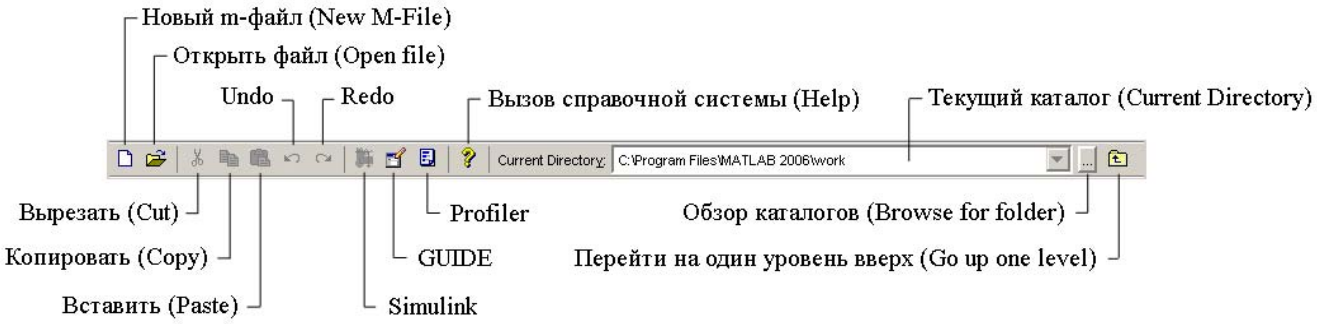

Рис. 20. Стандартная панель инструментов **Desktop Toolbar**

Кнопка **Simulink** обеспечивает вызов самого популярного расширения пакета MATLAB – системы имитационного моделирования. Кнопка **GUIDE** вызывает конструктор интерфейса, с помощью которого в диалоговом режиме формируется интерфейс приложения. В правой части панели расположен раскрывающийся список, позволяющий пользователю выбрать текущий рабочий каталог (**Current Directory**)и две кнопки, обеспечивающие просмотр каталогов (**Browse For Folder**) и переход по каталогам на один уровень вверх (**Go Up One Level**).

1.4.2. Пользовательская панель инструментов. Пользовательская панель инструментов (**Shortcuts Toolbar**) (рис. 21), как понятно из названия, позволяет пользователю добавлять свои наиболее часто используемые элементы.

Рис. 21. Пользовательская панель инструментов **Shortcuts Toolbar**

1.4.3. Панель инструментов окна Current Directory. Окно **Current Directory** (Текущий каталог) также содержит инструментальную панель (рис. 22), на которую вынесено несколько кнопок, ускоряющих выполнение некоторых операций по обзору, управлению и поиском папок и файлов и снятию окна с якоря.

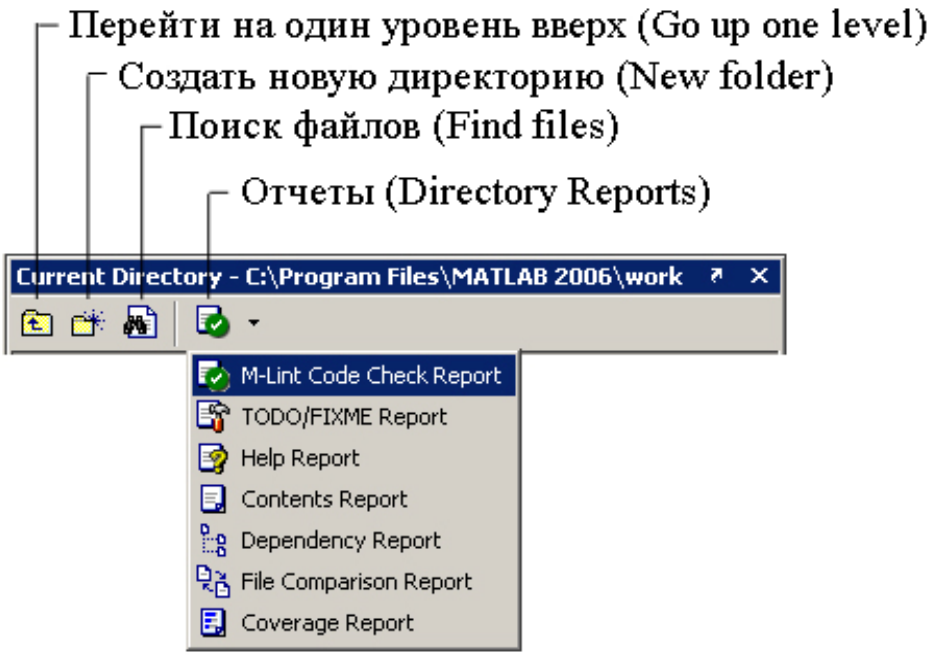

Рис. 22. Панель инструментов окна **Current Directory**

1.4.4. Панель инструментов окна Workspace. На панель окна **Workspace** (Рабочее пространство) тоже вынесено несколько кнопок (рис. 23), ускоряющих выполнение некоторых операций по сохранению и восстановлению переменных рабочего пространства, по очистке рабочего пространства и снятию окна с якоря.

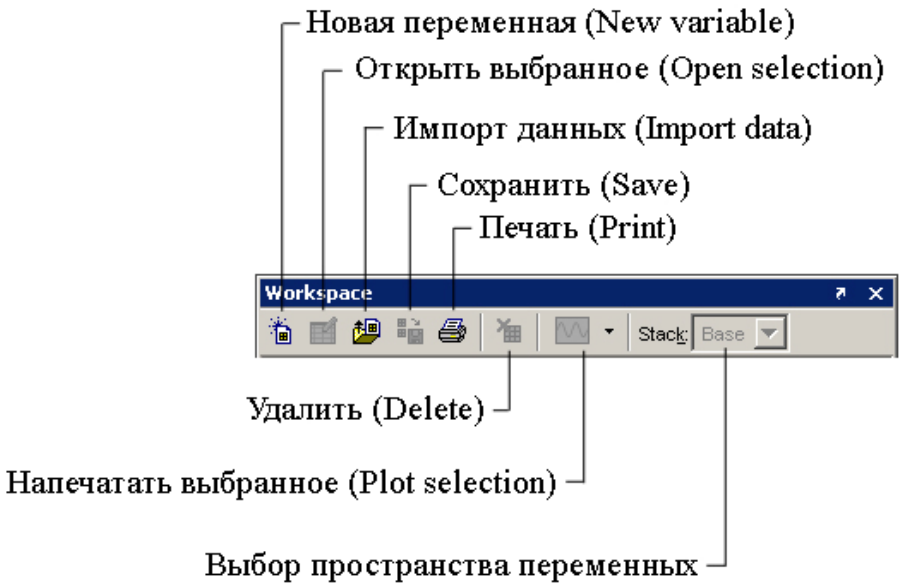

Рис. 23. Панель инструментов окна **Workspace**

1.5. Настройка среды MATLAB

Настройка параметров системы доступна через диалоговое окно **Preferences**, представленное на рис. 24. Это окно разделено на две части: в левой части окна расположен иерархический раскрывающийся список категорий настроек; в правой части окна отображаются доступные для изменения параметры выбранной категории настроек.

Далее рассмотрим наиболее часто используемые настройки ссреды.

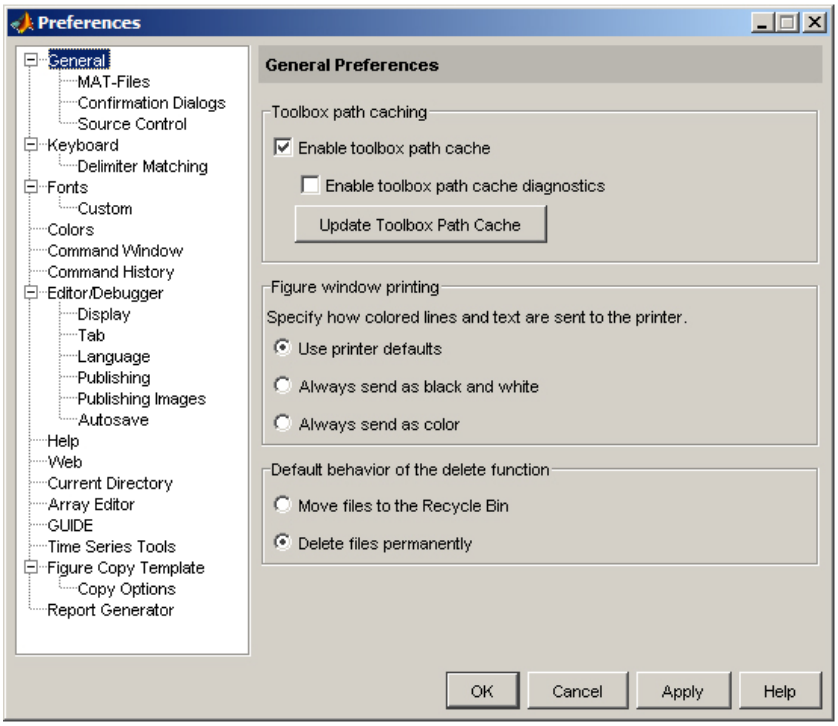

Рис. 24. Окно **Preferences**

1.5.1. Настройки сохранения mat-файлов. Данная категория настроек содержит всего одну группу параметров – **MAT-file save options.** Данные настройки применяются при использовании команды **Save**. По умолчанию (**Use Default Features**) MATLAB при сохранении сжимает данные и использует перекодировку символов в формате Unicode. При выборе второго варианта сохранения mat-файлов (**Ensure Backward Compatibility**) обеспечивается совместимость с предыдущими версиями MATLAB (до 6-й), поскольку в них еще не была реализована возможность перекодировки символов в формате Unicode при сохранении.

1.5.2. Подтверждающие диалоговые окна. С помощью подтверждающих диалоговых окон **Confirmation Dialogs** (рис 25) пользователь может установить или отменить появление соответствующего подтверждающего диалогового окна при выполнении одного из действий, приведенных в списке ниже (в скобках указано имя инструментального окна):

- удаление элемента истории команд (**Command History**);
- очистка командного окна (**Command Window**);
- попытка редактирования несуществующего файла (**Editor**);
- сохранение при активации (**GUIDE**);
- сохранение при экспорте (**GUIDE**);
- при выходе из MATLAB (**General**);
- при удалении переменных (**Workspace**).

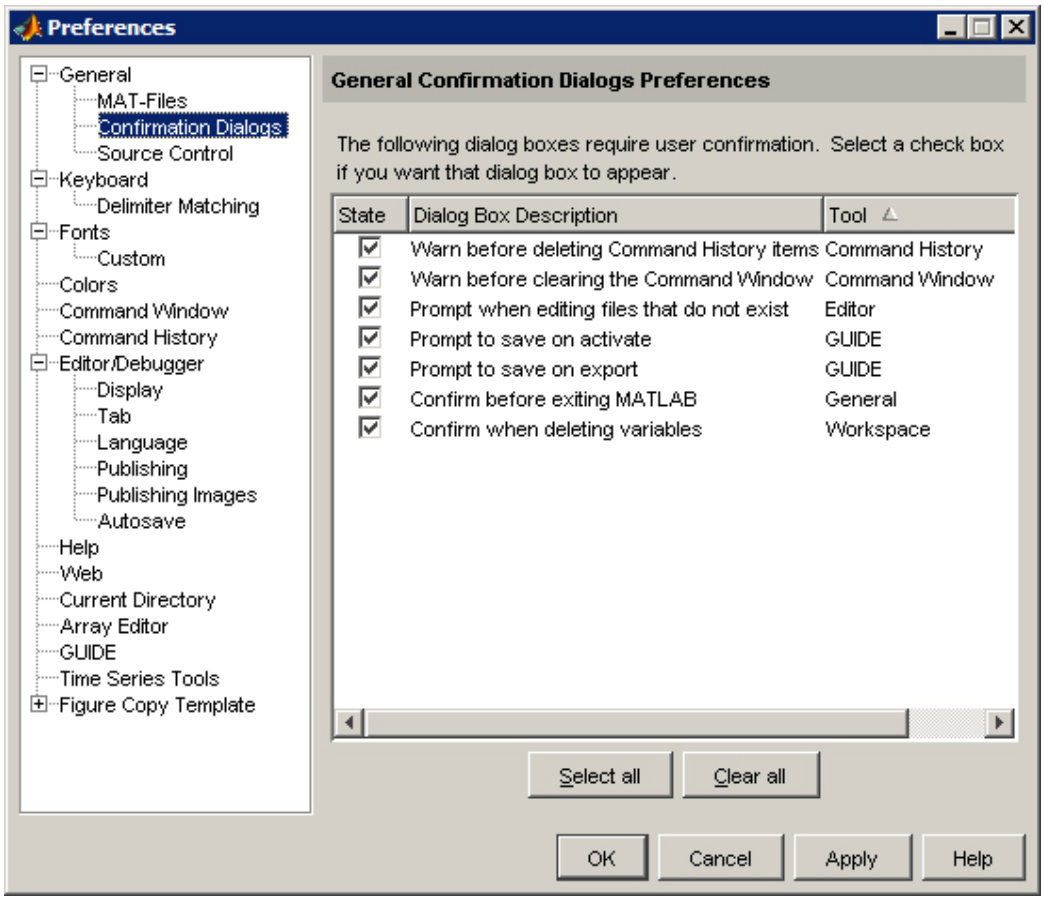

Рис. 25. Подтверждающие диалоговые окна

1.5.3. Параметры настройки шрифтов. На рис. 26 приведены настройки параметров шрифтов.

Первая группа опций **Desktop code font** позволяет настроить параметры отображения шрифтов в следующих инструментальных окнах: командном окне (**Command Window**), окне истории команд (**Command History**) и окне редактора исходных кодов (**Editor**).

Следующая группа опций **Desktop text font** позволяет настроить параметры отображения шрифтов в окне навигатора справочной системы (**Help Navigator**), окне выбора текущего рабочего каталога (**Current Directory**), окне рабочего пространства переменных (**Current Directory**) и окне редактора массивов (**Array Editor**).

Использование опции **Use antialiasing to smooth desktop fonts** позволяет установить или отменить более гладкое отображение шрифтов.

Категория настройки шрифтов **Fonts** содержит подкатегорию **Custom** (рис. 27). Данная подкатегория позволяет осуществить произвольную настройку параметров шрифтов для любого инструментального окна, указанного в списке **Desktop tools**. В данном случае возможна настройка индивидуальных шрифтовых параметров, таких как используемый шрифт, его начертание (жирный шрифт, наклонный шрифт и т. д.) и высота.

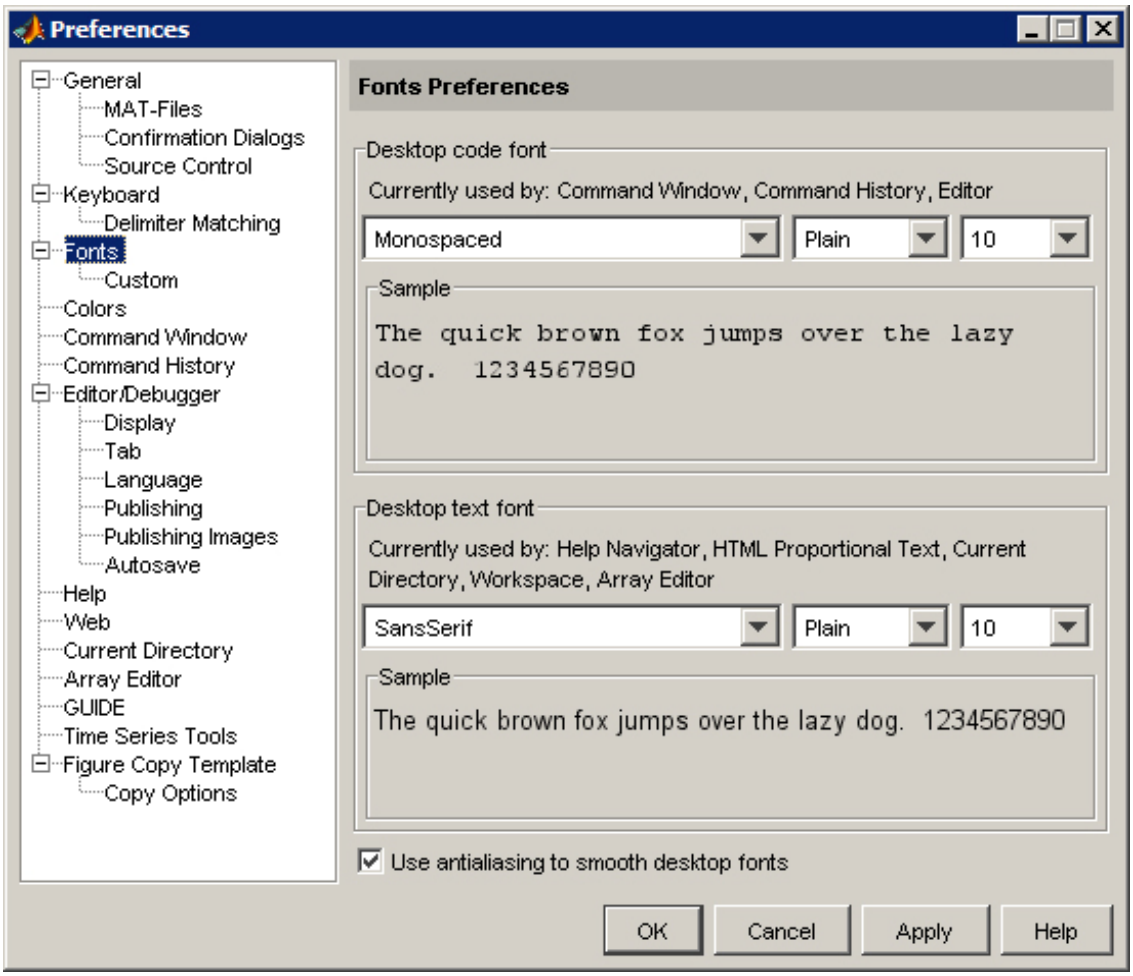

Рис. 26. Параметры настройки шрифтов

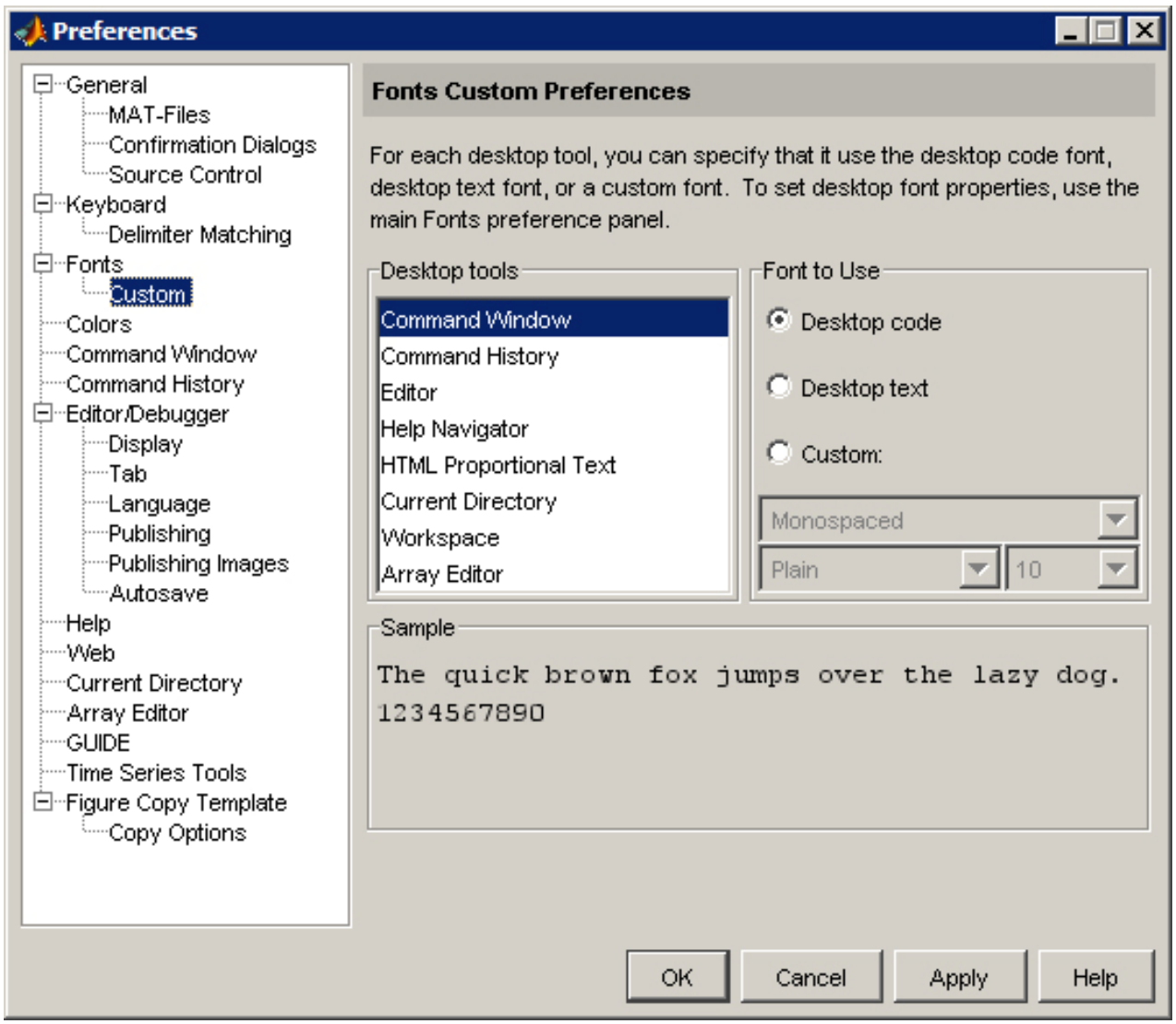

Рис. 27. Опции для произвольной настройки шрифтов

1.5.4. Параметры настройки цветов. На рис. 28 приведены параметры настройки цветов.

Группа опций **Desktop tool colors** позволяет настроить параметры цвета для текста и фона всех инструментальных окон MATLAB. По умолчанию предполагается использование системных цветов (установленный флажок **Use system colors**), т. е. цвет текста – черный, цвет фона – белый, но, убрав флажок **Use system colors**, можно настроить эти цвета в соответствии со своими предпочтениями.

Группа опций **M-file syntax highlighting colors** позволяет настроить параметры цвета для текста и фона в редакторе исходных кодов **Editor** при редактировании m-файлов. В данном случае доступны индивидуальные настройки цветов для ключевых слов (**Keywords**), комментариев (**Comments**), строк (**Strings**), незавершенных строк (**Unterminated strings**), системных команд (**System commands**) и ошибок (**Errors**).

В правой нижней части окна расположена кнопка **Restore Default colors**, при нажатии на которую, восстанавливаются исходные настройки цветов.

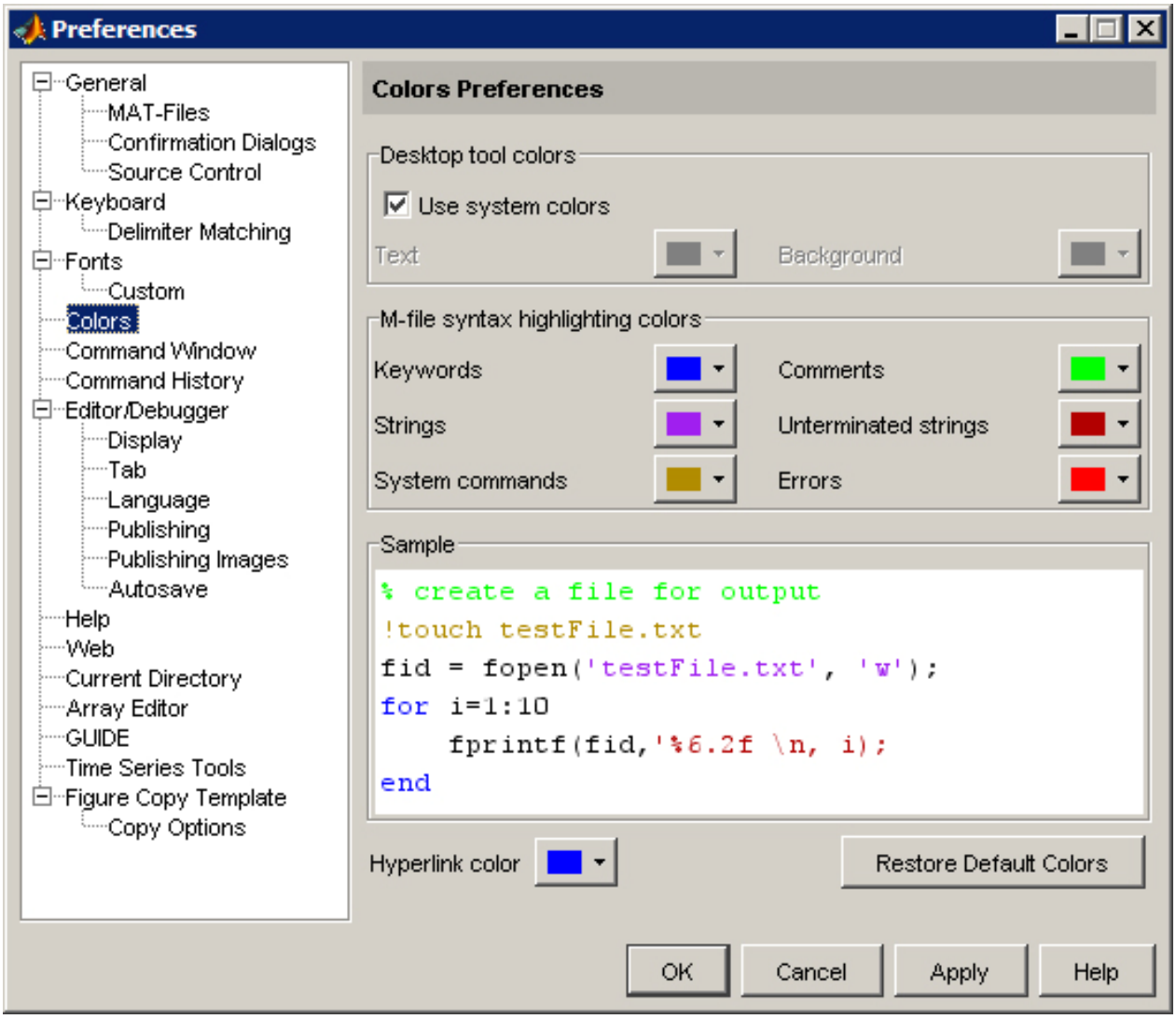

Рис. 28. Параметры настройки цветов

1.5.5. Параметры настройки окна Command Window. Параметры настройки окна **Command Window** приведены на рис. 29.

Группы настроек **Text display** и **Display** позволяет пользователю установить параметры отображения текста в командном окне:

- формат отображения чисел (**Numeric format**),
- межстрочный интервал (**Numeric display**),
- автоматический переход на новую строку при длине набираемой строки больше ширины окна (**Wrap lines**),
- ограничение на количество отображаемых по ширине экрана столбцов матриц до 80-ти (**Limit matrix display width to eighty columns**),
- размер буфера строк **Number of lines in command window scroll buffer** (максимальный размер 25000 строк).

Опция **Tab size** позволяет установить длину символа табуляции (**Tab key**) в пробелах.

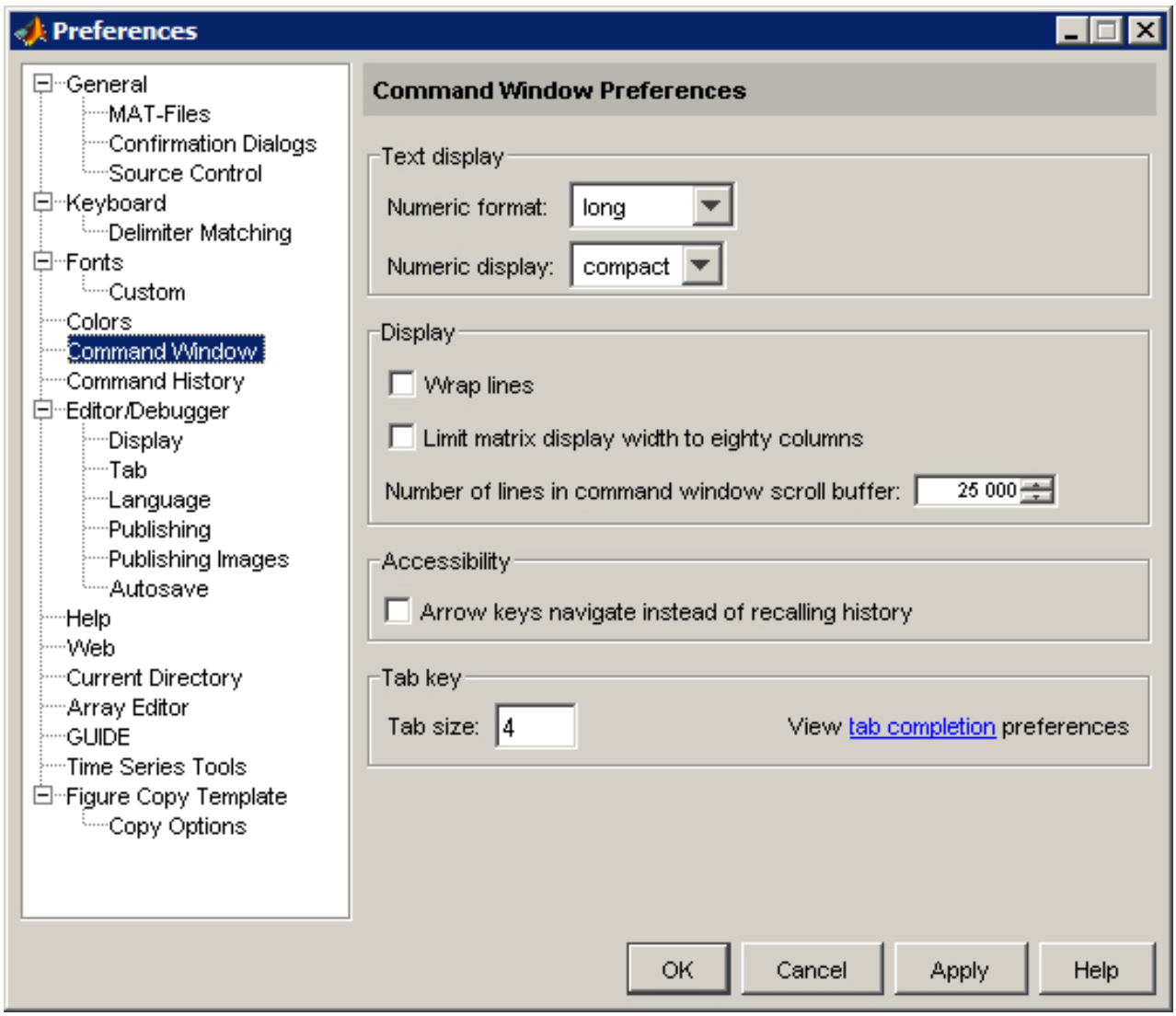

Рис. 29. Опции для настройки параметров окна **Command Window**

1.6. Редактор исходных кодов Editor

Начиная с версии 6.0, в пакете MATLAB появился мощный специализированный редактор исходных кодов **Editor** (рис. 30). За более чем 5 лет своего существования он претерпел значительное количество изменений, и теперь является наиболее удобным средством написания и редактирования программ на языке MATLAB.

По аналогии с рабочей средой основными элементами **Editor** являются:

- выпадающие меню;
- стандартная панель инструментов с кнопками и раскрывающимся списком;
- панель для управления ячейками (**Toolbar for Cell Features**)
- окно для ввода текста;
- поле для отображения нумерации строк;
- полу для отображения установленных точек останова (**Breakpoints**);
- строка состояния.

1.7. Команды общего назначения

Данные команды набираются с клавиатуры после знака приглашения (>>). Выполнение команд следует после нажатия клавиши <Enter>. Команды разделены на следующие группы:

- управляющие команды и функции;
- команды управления переменными и рабочим пространством;
- команды работы с файлами и операционной системой;
- команды управления командным окном;
- команды запуска и выхода из MATLAB;
- команды получения общей информации.

Ниже приведены краткие характеристики команд каждой группы.

Управляющие команды и функции:

help	Вывод на экран первых строк описания указанной программы или функции
what	Вывод на экран перечня имен m-, mat- и mex-файлов в текущем каталоге
type	Вывод на экран текста указанного m-файла
lookfor	Поиск программы (функции) по указанному ключевому слову
which	Вывод на экран полного пути местонахождения указанной функции или файла
demo	Запуск программы демонстрации возможностей MATLAB
path	Вывод на экран полного перечня путей поиска файлов MATLAB по умолчанию

Команды управления переменными и рабочим пространством:

who	Вывод на экран перечня текущих переменных
whos	Расширенная форма перечня текущих переменных
load	Загрузка в рабочее пространство значений переменных из указанного файла на диске
save	Запись значений переменных рабочего пространства в указанный файл на диске
clear	Очистка памяти от переменных и функций
pack	Уплотнение памяти рабочего пространства
size	Определение размеров массива
length	Определение длины одномерного массива
disp	Вывод на экран матрицы или текста

Команды работы с файлами и операционной системой:

cd	Заменить текущий каталог на указанный
dir	Вывести на экран листинг указанного каталога
open	Открыть указанный файл
delete	Уничтожить (стереть) указанный файл
getenv	Вывести значения параметров окружения (среды)

!	Выполнить как команду операционной системы (применяется после указания команды операционной системы)
unix	Выполнить как команду операционной системы и вывести результат
diary	Записать текст командного окна в дневник MATLAB

Команды управления командным окном:

cedit	Установить командную строку редактора клавиш
clc	Очистить командное окно
home	Поместить курсор в начало страницы
command window	Открыть окно соманд или вывести его на передний план, если оно уже было открыто ранее
format	Установить указанный формат вывода чисел на экран
echo	Установка или упразднение режима эхо-печати текста выполняемой программы
more	Установка режима постраничного вывода текста в командное окно

Команды запуска и выхода из MATLAB:

matlabrc	Запуск главного стартового m-файла
startup	Запуск MATLAB через m-файл **startup**
quit (exit)	Выйти из MATLAB
finish	m-файл с командами, выполняемыми при выходе из MATLAB

Команды получения общей информации:

info	Получение информации о MATLAB и фирме MathWorks, Inc.
subscribe	Подписка по Internet как пользователя MATLAB
whatsnew	Информация о новых возможностях, которые не вошли в документацию
version	Информация о версии MATLAB
ver	Информация о версии MATLAB и версиях установленных компонентов (SIMULINK, TOOLBOX)

2. ПРОСТЕЙШИЕ ВЫЧИСЛЕНИЯ

2.1. MATLAB в роли суперкалькулятора

Система MATLAB создана таким образом, что любые, даже весьма сложные вычисления можно выполнять в режиме прямых вычислений (в режиме калькулятора), не прибегая к составлению программы. В этом режиме MATLAB способен производить не только обычные для калькулятора вычисления (например, выполнять арифметические операции и вычислять элементарные функции), но и операции с векторами и матрицами, комплексными числами, рядами и полиномами. Можно, практически, мгновенно задать и вывести графики различных функций – от простой синусоиды до сложной трехмерной фигуры.

Работа с системой MATLAB в режиме прямых вычислений носит диалоговый характер. Пользователь набирает на клавиатуре вычисляемое выражение, редактирует его (если нужно) в командной строке и завершает ввод нажатием клавиши <Enter>, после чего получает немедленный ответ. Математические выражения составляют основу всех математических программных систем и пакетов и строятся с помощью чисел, констант, переменных, операторов и функций в сочетании с различными спецзнаками. Примеры простых математических выражений:

```
2+3,   6.402*sin(x),   0.2+exp(2*t)/3,   sqrt(y^2)/5,   cos(pi/2)
```

2.2. Числа, константы и системные переменные

Число – простейший объект MATLAB, представляющий количественные данные. Ввод чисел с клавиатуры производится по общим правилам, принятым для языков программирования высокого уровня. При вводе целая часть числа отделяется от дробной с помощью точки. Так, если ввести в командном окне MATLAB строку

```
>> 2.17069341e-17
```

то после нажатия клавиши <Enter> в этом окне появится следующая запись

```
ans =
     2.1607e-017
```

Результат выводится по укороченному формату чисел (**Short**), применяемому системой по умолчанию. Требуемый формат устанавливается при вхождении в область с названием **Numeric Format** с помощью команды **Preferences** из меню **File** (см. п.1.2) либо непосредственно с клавиатуры при наборе команды **format** <name> (например, **format long e**).

При решении многих инженерных и научных задач приходится пользоваться элементарными математическими функциями, аргументы которых являются комплексными числами. В этом случае и результаты вычислений нередко выражаются в комплексном виде. Синтаксис комплексного числа записывается так: $z = \mathrm{Re}(z)+i*\mathrm{Im}(z)$ или в тригонометрической форме $z = r*(\cos\varphi+i*\sin\varphi)$, где $\mathrm{Re}(z)$, $\mathrm{Im}(z)$ – соответственно действительная и мнимая части числа; r и φ – модуль и фаза (значения аргумента φ в радианах от $-\pi$ до $+\pi$). Мнимая часть имеет множитель i или j, означающий корень квадратный из -1:

```
3+2i,     -1+j,     6.28i,     123.456+3.6e-3i
```

Функции **real(z)** и **imag(z)** возвращают соответственно действительную – $\mathrm{Re}(z)$ и мнимую – $\mathrm{Im}(z)$ части комплексного числа z. Для получения модуля r и фазы φ комплексного числа используются функции **abs(z)** и **angle(z)**. Функция **conj(z)** определяет число, комплексно сопряженное по отношению к исходному комплексному числу z. Ниже даны простейшие примеры работы с комплексными числами:

```
>> i                          >> j
ans =                         ans =
     0+1.0000i                     0+1.0000i

>> z = 2+3i                   >> conj(z)
ans =                          ans =
     2.0000+3.0000i                2.0000-3.0000i

>> real(z)                    >> imag(z)
ans =                         ans =
     2                             3

>> abs(z)                     >> angle(z)
ans =                         ans =
     3.6056                        0.9828
```

В арифметических выражениях языка MATLAB применяются следующие знаки операций:

+ сложение;
- вычитание;
* умножение;
/ деление слева направо;
\ деление справа налево;
^ возведение в степень.

Использование системы в режиме калькулятора может происходить путем простой записи в командную строку последовательности арифметических действий с числами, например:

```
>> 6.25*3.42^2 - 12.6/3.14
ans =
     69.0898
```

В различных форматах при повторении тех же действий получим следующие результаты:

format short	69.0898
format short e	6.9090e+001
format long	69.08976114649681
format long e	6.908976114649681e+001
format bank	69.09

Если запись оператора не заканчивается символом «;» (точка с запятой), то результат действия этого оператора сразу же выводится в командное окно. Если оператор заканчивается знаком «;», то результат не отображается в командном окне, т.е. знак «;» подавляет печать результата вычислений.

Если оператор не содержит знака присваивания «=», то MATLAB присваивает специальную системную переменную с именем **ans** (**ans**wer – ответ) при выводе результатов вычислений на экран. Полученное значение можно использовать в

последующих вычислениях под именем **ans**, однако, необходимо помнить, что значение системной переменной **ans** изменяется после действия очередного оператора без знака присваивания (о системных переменных см. ниже).

Заметим, что численный результат можно выводить без имени **ans** (или имени какой-либо другой переменной), если воспользоваться функцией **disp** (от слова «дисплей»), которая позволяет выводить в командное окно результаты вычислений или некоторый текст. Вывод результата вычисления арифметического выражения, рассмотренного в предыдущем примере, теперь можно осуществить так:

```
>> disp(6.25*3.42^2 - 12.6/3.14)
  69.0898
```

Константа – это предварительно определенное числовое (или символьное) значение, представленное именем. В MATLAB существует разновидность констант, которые называются *системными переменными*. Они снабжены именем и зарезервированы системой, так как задаются при ее загрузке. Основные системные переменные, применяемые в MATLAB, указаны ниже:

i или **j** – мнимая единица (корень квадратный из -1);
pi – число «пи»: 3.1415926…;
eps – погрешность для операций 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 не числовой характер данных (**N**ot-**a**-**N**umber).

Ниже даны некоторые примеры на применение системных переменных:

```
>> 2*pi
ans =
     6.2832

>> eps
ans =
     2.2204e-016

>> disp(realmin)
  2.2251e-308

>> disp(realmax)
  1.7977e+308

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

>> disp(0/0)
Warning: Divide by zero.
   NaN
```

Из приведенных примеров следует, что:

- введенные числа и результаты всех вычислений в системе MATLAB сохраняются в памяти компьютера с относительной погрешностью (**eps**) около 2.0e-16 (т. е. с точными значениями в 15 десятичных разрядах);

- диапазон представления модуля действительных чисел лежит в промежутке

между 10^{-308} и 10^{+308};

- в последних двух примерах система предупреждает (**Warning**) о том, что произведена операция деления на нуль (**Divide by zero**), вследствие чего в первом случае результат бесконечен (происходит переполнение разрядной сетки), а во втором – не является числом.

Результат, соответствующий бесконечно большому значению (**Inf**), может быть получен и другим путем, например, при вычислении экспоненты от большого аргумента: e^{900}. Записывая экспоненциальную функцию (**exp**), имеем:

```
>> exp(900)
ans =
     Inf
```

Точно также сообщение системы в случае результата, не представимого в виде числа (**NaN**), может следовать при многих операциях. Ниже приведены основные из этих операций: умножение вида **0*Inf**; деление вида **0/0** и **Inf/Inf**; сложение и вычитание бесконечностей, например: **(Inf) - (-Inf)**; любые арифметические операции с переменной **NaN** и т. д.

Символьные константы – это цепочка символов, заключенных в апострофы, например:

```
'2+3',     'СТРОКА',     'Hello my friend'.
```

Если в апострофы помещено арифметическое выражение, то оно *не вычисляется* и рассматривается как цепочка символов. Однако в MATLAB для подобных случаев предусмотрены специальные функции преобразования, с помощью которых символьные выражения могут быть преобразованы в вычисляемые.

Символьные выражения (например, текстовую информацию) удобно выводить в командное окно также с помощью функции **disp**, как это было показано ранее при выводе числовой информации. Например:

```
>> disp('Задача решена')
Задача решена
```

Текстовые комментарии. Поскольку MATLAB используется для достаточно сложных вычислений, то важное значение имеет наглядность их описания. Она достигается использованием *текстовых комментариев*, которые вводятся с помощью символа %. Этот символ является указателем того, что следующий за ним текст в данной строке системой игнорируется. В текстовых комментариях могут использоваться любые символы, в том числе буквы русского алфавита. Комментарии предназначены исключительно для пользователя и чаще всего используются в текстах m-файлов для описания тех или иных операций, поясняющих работу программы, например:

```
% Ниже представлено вычисление экспоненциальной функции
```

2.3. Переменные и оператор присваивания

Переменные это объекты MATLAB, наделенные именем. Они способны хранить некоторые, обычно разные по значению данные. В зависимости от этих данных переменные могут быть числовыми или символьными, векторными или матричными.

Задание определенных значений переменных выполняется через операцию присваивания. Оператор присваивания имеет вид:

<имя переменной> = <выражение> [;]

Здесь и далее при записи синтаксиса оператора выражение в квадратных скобках обозначает необязательный элемент конструкции оператора. Типы переменных заранее не декларируются. Они определяются выражением, значение которого присваивается переменной. Если это выражение вектор или матрица, то переменная будет векторной или матричной.

Имя переменной (ее идентификатор) это любая последовательность латинских букв, цифр и символа подчеркивания «_», начинающаяся с буквы (напр., A, X1, balka, Alpfa_2 и т. д.).

Операторы состоят из специальных символов, имен функций и переменных, а также числовых констант и могут оканчиваться запятой или точкой с запятой, которые управляют выводом результата на экран. Если в одной строке содержится несколько операторов, то при их записи необходимо использовать разделители. После единственного или последнего оператора строки разделитель можно не ставить.

Длина буфера командной строки ограничена 4096 символами (в ранних версиях – 256 символами). При сложном виде оператора для его ввода может быть не достаточно одной строки. Чаще всего это происходит тогда, когда математическое выражение длинное и все его символы не помещаются в видимой области окна. Тогда часть выражения с целью удобства и наглядности записи оператора можно перенести на новую строку с помощью многоточия из трех точек), например:

```
s = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 - 1/8 + ...
   1/9 - 1/10 + 1/11 - 1/12 + 1/13 - 1/14 + 1/15;
```

Ниже приведено задание переменных и их уничтожение в рабочем пространстве с помощью функции **clear**:

```
>> a = pi/2; b = sin(a)
b =
   1

>> clear b, c = a+2, a = pi^2, b
c =
   3.5708



a =
```

```
   9.8696
??? Reference to a cleared variable b.

>>clear

>>a
??? Reference to a cleared variable a.


>>c
??? Reference to a cleared variable c.
```

Во втором примере выборочно стерта переменная *b* по команде **clear b**, а переменной *a* присвоено новое значение. После уничтожения переменной *b* она становится неопределенной, поэтому при обращении к ней система выдает сообщение об ошибке. По команде **clear** без параметров стерты все остальные переменные (*a* и *c*).

2.4. Элементарные математические функции

Общая форма вызова функции MATLAB имеет следующий вид:

<имя переменной> = <имя функции>(<список входных величин>)

В языке MATLAB предусмотрены встроенные элементарные функции, включающие тригонометрические, гиперболические, экспоненциальные и логарифмические функции, а также функции для работы с комплексными числами и для округления чисел различными способами.

Тригонометрические и гиперболические функции:

sin(X), cos(X)	синус, косинус числа **X**;
tan(X), cot(X)	тангенс, котангенс;
sec(X), csc(X)	секанс, косеканс;
asin(X), acos(X)	арксинус, арккосинус;
atan(X), acot(X)	арктангенс, арккотангенс;
asec(X), acsc(X)	арксеканс, арккосеканс;
sinh(X), cosh(X)	гиперболические синус и косинус;
tanh(X), coth(X)	гиперболические тангенс и котангенс;
asinh(X)), acosh(X)	гиперболические арксинус и арккосинус;
atanh(X), acoth(X)	гиперболические арктангенс и арккотангенс.

Замечание: Аргументы тригонометрических функций должны быть выражены в радианах. Обратные тригонометрические функции возвращают результат также в радианах.

Экспоненциальная функция, логарифмы, степенные функции:

exp(X)	экспонента числа **X**;
log(X)	натуральный логарифм;
log10(X)	десятичный логарифм;

log2(X) логарифм по основанию 2;
pow2(X) возведение числа 2 в степень **X**: 2^X;
sqrt(X) квадратный корень из числа **X**.

Функции для работы с комплексными числами. Перечень, краткое описание и примеры использования данных функций даны в п. 2.2.

Округление и остаток от деления:
fix(X) округление до ближайшего целого в сторону нуля;
floor(X), ceil(X) округление до ближайшего целого в сторону -∞, +∞;
round(X) округление до ближайшего целого;
mod(X) остаток от целочисленного деления (со знаком);
rem(X, Y) вычисление остатка от целочисленного деления **X** на **Y**;
sign(X) вычисление сигнум-функции числа **X** (0 при **X** = 0, -1 при **X** < 0, +1 при **X** > 0).

Ниже приведены примеры на использование этих функций в MATLAB:

```
>> fix(1.8)
ans =
     1

>> fix(-1.9)
ans =
     -1

>> floor(3.2)
ans =
     3

>> ceil(3.2)
ans =
     4

>> round(4.1)
ans =
     4

>> round(4.5)
ans =
     5

>> mod(5, 2)
ans =
     1

>> mod(5, -2)
ans =
     -1

>> rem(5, 2)
ans =
     1

>> rem(5, -2)
ans =
     1

>> sign(-15)
ans =
     -1

>> sign(0)
ans =
     0
```

2.5. Специальные математические функции

Кроме элементарных функций в MATLAB имеются встроенные специальные математические функции, такие как функции Бесселя, полиномы Лежандра и т. д. Ниже приведен перечень и краткое описание некоторых из этих функций. Правила вызова и использования функций можно найти в их описаниях, которые выводятся на экран, если набрать команду **help** <name>, где *name* – имя функции или

обратившись к интерактивной справочной системе MATLAB.

Функции преобразования координат:

cart2sph преобразование декартовых координат в сферические;
cart2pol преобразование декартовых координат в полярные;
pol2cart преобразование полярных координат в декартовые;
sph2cart преобразование сферических координат в декартовые.

Функции Бесселя:

besselj функция Бесселя первого рода;
bessely функция Бесселя второго рода;
besseli модифицированная функция Бесселя первого рода;
besselk модифицированная функция Бесселя второго рода.

Бета-функции:

beta бета-функция;
betainc неполная бета-функция;
betaln логарифм бета-функции.

Гамма-функции:

gamma гамма-функция;
gammainc неполная гамма-функция;
gammaln логарифм гамма-функции.

Эллиптические функции и интегралы:

ellipj эллиптические функции Якоби;
ellipke полный эллиптический интеграл;
expint функция экспоненциального интеграла.

Функции ошибок:

erf функция ошибок;
erfc дополнительная функция ошибок;
erfcx масштабированная дополнительная функция ошибок;
erfinv обратная функция ошибок.

Другие функции:

gcd наибольший общий делитель;
lcm наименьшее общее кратное;
legendre обобщенная функция Лежандра;
rat представление числа в виде рациональной дроби;
rats представление чисел в виде рациональной дроби.

3. РАБОТА С МАТРИЦАМИ И МАССИВАМИ

3.1. Основные определения и понятия

Для проведения вычислений с такими объектами как *векторы, матрицы* (*или тензоры*) необходим способ их хранения в компьютере. Для этой цели служат массивы, в виде которых представляются все данные MATLAB. В одномерных массивах хранятся векторы (в виде строк или столбцов), в двумерных – матрицы, в многомерных – тензоры.

Массив – упорядоченная, пронумерованная совокупность однородных данных, снабженная именем. Массивы различаются числом измерений (или размерности): одномерные, двумерные, многомерные. *Размер* массива – число элементов вдоль каждого из измерений. Доступ к элементам массива осуществляется при помощи *индекса* (напр., **A(k, i)**, где **k, i** – индексы двумерного массива **A**). Нумерация элементов массива в MATLAB всегда начинается с единицы, поэтому индексы не могут быть отрицательными или равными нулю.

3.2. Особенности задания векторов и матриц

MATLAB – система, специально предназначенная для выполнения сложных вычислений с векторами, матрицами и массивами. При этом она по умолчанию считает, что каждая переменная это вектор или матрица. Например, при задании **X = 1** система полагает, что **X** есть вектор, состоящий из одного элемента, значение которого = 1. Для задания вектора из нескольких элементов необходимо их значения перечислить в квадратных скобках, разделяя элементы пробелами или запятыми. Например:

```
>> X = [1 2 3]
X =
   1     2     3
```

```
>> X = [1, 2, 3]
X =
   1     2     3
```

В данном примере рассмотрен ввод вектора-строки **X,** содержащего 3 элемента. Причем представлены два варианта ввода вектора **X**. После нажатия клавиши <Enter> результаты выводятся на экран дисплея. Вектор-столбец вводится аналогично вектору-строке, но значения элементов в перечне отделяются знаком «;» (точка с запятой) либо путем использования операции транспонирования. Запись **X'** означает транспонирование вектора **X**. Например:

```
>> X = [1; 2; 3]
X =
   1
   2
   3
```

```
>> X = [1 2 3]
X =
   1
   2
   3
```

Применение знака **:** *(двоеточие) для формирования векторов.* Оператор **:** (двоеточие) используется в MATLAB для различных целей. В частности, он позволяет генерировать числовые последовательности, необходимые для создания векторов или значений абсциссы при построении графиков. Общая форма задания

такой последовательности имеет вид:

<начальное значение>:<шаг>:<конечное значение>

Шаг может быть как положительным, так и отрицательным. При шаге равном 1 он может опускаться в записи последовательности, в результате чего вышеприведенная конструкция принимает более упрощенный вид:

<начальное значение>:<конечное значение>

Примеры:

```
>> 1:6
ans =
     1   2   3   4   5   6

>> i = 0:2:10
i =
   0   2   4   6   8  10

>> j = 10:-1:4
j =
   10   9   8   7   6   5   4

>> V = 1:-0.2:0
V =
    1.0000   0.8000   0.6000   0.4000   0.2000   0

>> t = -2.1:5
t =
   Columns 1 through 5
   -2.1000  -1.1000  -0.1000   0.9000   1.9000
   Columns 6 through 8
    2.9000   3.9000   4.9000

>> X = 0:pi/2:2*pi
X =
   0   1.5708   3.1416   4.7124   6.2832

>> Y = sin(X)
Y =
   0   1.0000   0.0000  -1.0000   0.0000
```

Задание матрицы требует указания различных строк. Для различения строк используется знак «;» (точка с запятой). Например:

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

Возможен ввод элементов матриц и векторов в виде арифметических выраже-

ний, содержащих любые допустимые системе функции, например:

```
>> P = [2+3/(4+5) exp(-5); sqrt(10) 2.1*sin(3)]
P =
   2.3333   0.0067
   3.1623   0.2964
```

Для указания отдельного элемента вектора или матрицы используются выражения вида **X(i)** или **M(i, j)**. Так, задавая элемент матрицы *M* при *i* = 2 и *j* = 3, будем иметь:

```
>> M(2, 3)
ans =
     6
```

Можно, наоборот, любому элементу матрицы или вектора присвоить новое значение. Например, если на место элемента **M(3,3)** надо вставить число 10, то следует записать:

```
>> M(3, 3) = 10; M
M =
   1  2   3
   4  5   6
   7  8  10
```

Для задания векторов и матриц с комплексными элементами можно поступить так:

```
>> i = sqrt(-1);
>> M1 = [1 2; 3 4]+i*[5 6; 7 8]
M1 =
    1.000+5.000i   2.000+6.000i
```

или так:

```
>> i = sqrt(-1);
>> M1 = [1+5i 2+6i; 3+7i 4+8i]
M1 =
    3.000+7.000i   4.000+8.000i
```

Заметим, что оператор присваивания **i = sqrt(-1)** необходимо записывать в том случае, если перед этим в сеансе работы переменной *i* присваивалось какое-либо значение. В противном случае запись данного оператора необязательна, так как системная переменная *i* – мнимая единица по умолчанию.

3.3. Формирование векторов и матриц специального вида

MATLAB имеет несколько функций, которые позволяют формировать векторы и матрицы определенного (специального) вида. Ниже приведены некоторые из этих функций:

zeros(M, N) – создает матрицу размером (*M*×*N*) с нулевыми элементами:

```
>> zeros(4, 5)
ans =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

ones(M, N) – то же самое – с единичными элементами:

```
>> ones(4, 5)
ans =
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

eye(M, N) – создает матрицу размером ($M \times N$) с единичными элементами по главной диагонали и остальными нулевыми элементами:

```
>> eye(4, 5)
ans =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
```

Замечание: При формировании квадратных матриц записи данных функций допускают упрощения: **zeros(N)**, **ones(N)**, **eye(N)**. Если необходимо формировать специальные матрицы по размеру заданной матрицы A порядка ($M \times N$), то возможна форма записи: **zeros(size(A))**, **ones(size(A))**, **eye(size(A))**. Нулевой вектор-столбец задается с помощью функции **zeros(N, 1)**, а нулевая вектор-строка – с помощью функции **zeros(1, N)**. Аналогичным образом для создания соответствующих векторов используются функции **ones** и **eye**.

Рассмотрим еще несколько полезных команд:

rand(M, N) – создает матрицу размером ($M \times N$) из случайных чисел, равномерно распределенных в диапазоне от 0 до 1:

```
>> rand(3, 5)
ans =
    0.4175    0.9304    0.0920    0.7012    0.2625
    0.6868    0.8462    0.6539    0.9103    0.0475
    0.5890    0.5269    0.4160    0.7622    0.7361
```

randn(M, N) – создает матрицу размером ($M \times N$) из случайных чисел, распределенных по нормальному закону с нулевым математическим ожиданием и стан-

дартным (среднеквадратическим) отклонением, равным единице:

```
>> randn(3, 5)
ans =
      1.1650     0.3516    0.0591     0.8717     1.2460
      0.6268    -0.6965    1.7971    -1.4462    -0.6390
      0.0751     1.6961    0.2641    -0.7012     0.5774
```

Одна из интересных возможностей MATLAB – вычисление магического квадрата (функция **magic**). По команде **magic(N)** возвращается матрица размером ($N\times N$), состоящая из целых чисел от 1 до N^2, в которой суммы элементов по строкам, столбцам, главной и побочной диагоналям, соответствкнно, равны одному и тому же числу. При $N = 2$ построение магического квадрата невозможно. Пример:

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

Следующие функции **hadamard(N), hilb(N), invhilb(N), pascal(N)** – создают квадратные матрицы соответственно Адамара, Гильберта, обратную матрицу Гильберта и матрицу Паскаля размера ($N\times N$). О работе данных функций см. библиографию [1–3, 5] и справочную библиотеку MATLAB.

В языке MATLAB предусмотрено несколько функций, которые позволяют формировать матрицу на основе другой (заданной) матрицы или используя некоторый заданный вектор. К таким функциям относятся следующие:

fliplr(A) – формирует матрицу, переставляя столбцы заданной матрицы A относительно вертикальной оси, например:

```
A =
    1    2    3    4    5    6
    7    8    9   10   11   12
   13   14   15   16   17   18

>> fliplr(A)
ans =
     6    5    4    3    2    1
    12   11   10    9    8    7
    18   17   16   15   14   13
```

flipud(A) – формирует матрицу, переставляя строки заданной матрицы A относительно горизонтальной оси:

```
>> flipud(A)
ans =
    13   14   15   16   17   18
```

```
     7     8     9    10    11    12
     1     2     3     4     5     6
```

rot90(A) – формирует матрицу путем «поворота» заданной матрицы *A* на 90 градусов против часовой стрелки:

```
% Сравните эту операцию с операцией транспонирования A'
>> rot90(A)
ans =
     6    12    18
     5    11    17
     4    10    16
     3     9    15
     2     8    14
     1     7    13
```

repmat(A, p, q) – формирует матрицу, составленную из (*p*×*q*) копий заданной матрицы *A* порядка (*M*×*N*). При этом число элементов сформированной матрицы должно быть (*r*×*s*), где: *r* = *M*×*p*, *s* = *N*×*q*:

```
% Формирование 2×3 копии матрицы [1 2; 0 -8]
>> repmat([1 2; 0 -8], 2, 3)
ans =
     1     2     1     2     1     2
     0    -8     0    -8     0    -8
     1     2     1     2     1     2
     0    -8     0    -8     0    -8
```

reshape(A, p, q) – образует матрицу размером (*p*×*q*) путем последовательной выборки элементов заданной матрицы *A* порядка (*M*×*N*) по столбцам. При этом число элементов матрицы *A* и сформированной матрицы должно быть одинаковым, т. е. *M*×*N* = *p*×*q*:

```
% Матрица A (3×6) задана выше (см. функцию fliplr)
>> reshape(A, 2, 9)
ans =
     1    13     8     3    15    10     5    17    12
     7     2    14     9     4    16    11     6    18
```

tril(A) – образует нижнюю треугольную матрицу на основе матрицы *A* путем обнуления ее элементов выше главной диагонали:

```
>> tril(A)
ans =
     1     0     0     0     0     0
     7     8     0     0     0     0
    13    14    15     0     0     0
```

triu(A) – образует верхнюю треугольную матрицу на основе матрицы *A* путем обнуления ее элементов ниже главной диагонали:

```
>> triu(A)
ans =
      1    2    3    4    5    6
      0    8    9   10   11   12
      0    0   15   16   17   18
```

hankel(A) – образует квадратную матрицу Ганкеля, первые строка и столбец которой совпадают с заданным вектором *V*, например:

```
>> V = [3 5 -1 9];
>> hankel(V)
ans =
      3     5    -1     9
      5    -1     9     0
     -1     9     0     0
      9     0     0     0
```

Функция **diag(x)** в зависимости от вида входного параметра *x* выполняет две задачи: формирует или извлекает диагональ матрицы.

Формирование диагонали. Если *x – вектор*, то функция **diag(x)** создает квадратную матрицу с элементами заданного вектора на главной диагонали (другие элементы формируемой матрицы равны нулю). Так, используя введенный выше вектор *V*, получим:

```
>> diag(V)
ans =
      3     0     0     0
      0     5     0     0
      0     0    -1     0
      0     0     0     9
```

Заданный вектор можно формировать непосредственно в самой функции **diag**, используя для этой цели, например, приемы генерирования векторов (см. применение знака «:» в п. 3.2.) либо другие способы. В частности:

```
>> diag(-1:3)
ans =
     -1    0    0    0    0
      0    0    0    0    0
      0    0    1    0    0
      0    0    0    2    0
      0    0    0    0    3
```

```
>> diag(ones(1, 5))
ans =
      1    0    0    0    0
      0    1    0    0    0
      0    0    1    0    0
      0    0    0    1    0
      0    0    0    0    1
```

Заметим, что операция по формированию единичной матрицы, приведенная в последнем примере: **diag(ones(1,5))**, эквивалентна действию, определяемому с помощью функции **eye(5)**.

Элементы заданного вектора *x* могут быть установлены на любую (не обязательно главную) диагональ формируемой матрицы. Для этого при обращении к функции необходимо ввести еще один параметр *k* – целое число. Это число указывает на номер диагонали, отсчитываемый от главной диагонали. При этом от-

счет ведется вверх для положительного числа ($k > 0$) и – вниз для отрицательного ($k < 0$), например:

```
>> diag(V, -1)
ans =
     0     0     0     0     0
     3     0     0     0     0
     0     5     0     0     0
     0     0    -1     0     0
     0     0     0     9     0
```

```
>> diag(ones(3, 1), 2)
   ans =
        0     0     1     0     0
        0     0     0     1     0
        0     0     0     0     1
        0     0     0     0     0
        0     0     0     0     0
```

Извлечение диагонали. Если *x – матрица*, то функция **diag(x)** создает вектор-столбец, состоящий из элементов главной диагонали заданной матрицы *x*. Так, для матрицы *A*, введенной ранее в примере применения процедуры **fliplr**, извлечение главной диагонали осуществляется по следующей команде:

```
>> diag(A)
ans =
      1
      8
     15
```

Если из заданной матрицы необходимо извлечь какую-либо другую диагональ, то номер этой диагонали необходимо дополнительно указать в списке параметров для получения соответствующего вектор-столбца, например:

```
>> diag(A, 3)
ans =
      4
     11
     18
```

```
 >> diag(A, -2)
ans =
     13
```

В качестве примера, иллюстрирующего богатые возможности системы с встроенными функциями MATLAB, приведем оператор, позволяющий формировать ленточную трехдиагональную матрицу размера $2n+1$:

```
diag(-n:n) + diag(ones(2*n,1),1) + diag(ones(2*n,1),-1)
```

Замечание: При записи данного оператора использована операция сложения трех матриц **diag(-n:n), diag(ones(2*n, 1), 1)** и **diag(ones(2*n, 1), -1).** О сложении матриц и других действиях над ними см. п. 3.5.

При $n = 3$ и соответствующих действиях на клавиатуре, имеем:

```
>> n=3; diag(-n:n) + diag(ones(2*n,1),1) + diag(ones(2*n,1),-1)
ans =
    -3     1     0     0     0     0     0
     1    -2     1     0     0     0     0
```

```
     0     1    -1     1     0     0     0
     0     0     1     0     1     0     0
     0     0     0     1     1     1     0
     0     0     0     0     1     2     1
     0     0     0     0     0     1     3
```

3.4. Конкатенация матриц, удаление и вставка частей матриц

В процессе решения тех или иных задач часто возникает необходимость внесения некоторых изменений в исходные матрицы. Эти изменения могут касаться как простых операций, связанных, например, с заменой части элементов матрицы на их новые значения, так и более сложных преобразований. Часто требуется из нескольких малых матриц сформировать большую матрицу. Такое объединение малых матриц в большую называется *конкатенацией* (сцеплением). Иногда, напротив, требуется удалить из исходной матрицы часть строк или столбцов, в результате чего создается меньшая матрица из большей. Рассмотрим возможности системы MATLAB в отношении указанных операций.

Объединение малых матриц в большую (*конкатенация*)

Общий принцип горизонтальной конкатенации: блоки-матрицы A_i имеют одинаковое число строк и объединяются в обобщенную «строку»

$$A = [A_1, A_2, \ \dots\ , \ A_n],$$

отделяясь между собой пробелами или запятыми.

При вертикальной конкатенации блоки-матрицы A_i должны иметь одинаковое число столбцов, что позволяет сцепить их в обобщенный «столбец»

$$A = [A_1; A_2; \ \dots\ ; \ A_n].$$

При этом блоки отделяются друг от друга точкой с запятой.

Для иллюстрации конкатенации создадим вначале несколько исходных векторов и матриц:

```
>>A = ones(3)
A =
     1     1     1
     1     1     1
     1     1     1
```

```
>> B = eye(3)*(-3)
B =
    -3     0     0
     0    -3     0
     0     0    -3
```

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

```
>> X = [6:8]
X =
     6     7     8
```

```
>>Y = rand(2, 3)*10
Y =
    5.1942   0.3457   5.2970
```

```
>> Z = sin(Y)
Z =
   -0.8862    0.3389   -0.8339
```

```
   8.3097  0.5346  6.7115            0.8980   0.5095   0.4153

>> D = [A X' Y']  % Горизонтальная конкатенация
D =
   1.0000    1.0000    1.0000    6.0000    5.1942    8.3097
   1.0000    1.0000    1.0000    7.0000    0.3457    0.5346
   1.0000    1.0000    1.0000    8.0000    5.2970    6.7115

>> V = [A; B; X; Y; Z]  % Вертикальная конкатенация
V =
    1.0000     1.0000     1.0000
    1.0000     1.0000     1.0000
    1.0000     1.0000     1.0000
   -3.0000          0          0
         0    -3.0000          0
         0          0    -3.0000
    6.0000     7.0000     8.0000
    5.1942     0.3457     5.2970
    8.3097     0.5346     6.7115
   -0.8862     0.3389    -0.8339
    0.8980     0.5095     0.4153
```

Более сложный пример операции конкатенации:

```
>> U = [A X' B; C [Y; Z]]
U =
    1.0000   1.0000   1.0000   6.0000  -3.0000        0        0
    1.0000   1.0000   1.0000   7.0000        0  -3.0000        0
    1.0000   1.0000   1.0000   8.0000        0        0  -3.0000
   16.0000   2.0000   3.0000  13.0000   5.1942   0.3457   5.2970
    5.0000  11.0000  10.0000   8.0000   8.3097   0.5346   6.7115
    9.0000   7.0000   6.0000  12.0000  -0.8862   0.3389  -0.8339
    4.0000  14.0000  15.0000   1.0000   0.8980   0.5095   0.4153
```

Извлечение и вставка частей матриц

Для извлечения части матрицы из матрицы *A* необходимо обратиться к имени этой матрицы с указанием в круглых скобках номеров строк и столбцов следующим образом: **A(i:j, k:l)**. Пусть из полученной большой матрицы *U* необходимо выделить блок размером 4×4, ограниченный 2-й и 5-й строками и 3-м и 6-м столбцами:

```
>> M = U(2:5, 3:6)  % Извлечение подматрицы M из матрицы U
M =
    1.0000    7.0000         0   -3.0000
    1.0000    8.0000         0         0
    3.0000   13.0000    5.1942    0.3457
   10.0000    8.0000    8.3097    0.5346
```

Пусть требуется создать векторы X_1, и Y_1. Причем вектор X_1 состоит из элементов 2-го столбца матрицы *M*, а вектор Y_1 состоит из элементов 3-й строки матрицы *M*. Тогда необходимо записать:

```
>> X1 = M(:, 2)
X1 =
    7.0000
    8.0000
   13.0000
    8.0000
```

```
>> Y1 = M(3, :)
Y1 =
    3.0000   13.0000    5.1942    0.3457
```

Для вставки подматрицы *M* в большую матрицу *A* необходимо обратиться к имени этой матрицы с помощью следующего оператора присваивания: **A(i:j, k:l) = M**. При этом порядок подматрицы *M* должен соответствовать указанным в круглых скобках номерам строк и столбцов матрицы *A*. Пример:

```
% Вставка блока (4×2) из нулей в матрицу U
>> U(3:6, 2:3) = zeros(4, 2)
U =
    1.0000    1.0000    1.0000    6.0000   -3.0000         0         0
    1.0000    1.0000    1.0000    7.0000         0   -3.0000         0
    1.0000         0         0    8.0000         0         0   -3.0000
   16.0000         0         0   13.0000    5.1942    0.3457    5.2970
    5.0000         0         0    8.0000    8.3097    0.5346    6.7115
    9.0000         0         0   12.0000   -0.8862    0.3389   -0.8339
    4.0000   14.0000   15.0000    1.0000    0.8980    0.5095    0.4153
```

Отметим тонкие и неожиданные возможности системы MATLAB при выделении и вставке частей матриц. Рассмотрим пример:

```
>> A = [1:4; -5:3:4; 9:-2:3]  % Формирование матрицы A
A =
    1     2     3     4
   -5    -2     1     4
    9     7     5     3

>> M([3 1], :) = A(2:3, :)  % Замена двух строк матрицы M
M =
    9.0000    7.0000    5.0000    3.0000
    1.0000    8.0000         0         0
   -5.0000   -2.0000    1.0000    4.0000
   10.0000    8.0000    8.3097    0.5346
```

Заметим, что порядок замены задается с помощью индексов массива **M**, определяемых вектор-строкой [3 1]. Вначале заменяется 3-я строка матрицы *M* на 2-ю строку матрицы *A*, а затем 1-я строка *M* на 3-ю строку *A*.

Команда **A(:, n:-1:1)** переставляет столбцы *A* в обратном порядке; аналогичную операцию выполняет команда **A(n:-1:1, :)** со строками, например:

```
>> A(:, 4:-1:1)
ans =
     4     3     2     1
     4     1    -2    -5
```

```
     3     5     7     9
```

Любую матрицу *A* можно «растянуть» в единый вектор *V* с помощью простой записи: **V = A(:)**.

```
>> A = [1 2 3; 4 5 6]
A =
   1    2    3
   4    5    6
```

```
>> V = A(:)
V =
   1
   4
   2
   5
   3
   6
```

Удаление столбцов и строк матриц

Для удаления отдельных строк и столбцов матрицы используются пустые квадратные скобки **[ ]**. Проделаем это над матрицей *U*, заданной выше:

```
>> U(5:7, :) = []  % Удаление с 5-й по 7-ю строк матрицы U
U =
    1.0000   1.0000   1.0000   6.0000  -3.0000        0        0
    1.0000   1.0000   1.0000   7.0000        0  -3.0000        0
    1.0000        0        0   8.0000        0        0  -3.0000
   16.0000        0        0  13.0000   5.1942   0.3457   5.2970
```

Заметим, что эта операция эквивалентна записи: **U([5, 6 7], :) = [ ]**. Теперь из вновь образованной матрицы *U* удалим 2-й, 4-й и 7-й столбцы:

```
>> U([:, [2, 4, 7]) = []
U =
     1.0000    1.0000   -3.0000         0
     1.0000    1.0000         0   -3.0000
     1.0000         0         0         0
    16.0000         0    5.1942    0.3457
```

3.5. Операции с матрицами и массивами

В системе MATLAB реализовано два типа арифметических операций. Операции над матрицами определены в соответствии с правилами линейной алгебры, а операции над массивами выполняются поэлементно. Чтобы различать эти операции, операции над массивом предшествует точка. Операции сложения и вычитания над матрицами и массивами дают одинаковый результат. Все операции сведены в нижеследующую табл. 1.

Таблица 1

Знак	Операция	Условия выполнения операции

+	Сложение: **A + B**	Слагаемые должны быть одинакового размера или одно из слагаемых (любое) может быть скаляром. При этом скаляр добавляется ко всем элементам другого операнда. Данная операция однотипна как для матриц, так и для массивов
-	Вычитание: **A - B**	Величины *A*, *B* должны быть одинакового размера либо одна из них (любая) – скаляр; действия со скаляром выполняются по аналогии с операцией сложения. Данная операция однотипна как для матриц, так и для массивов
*	Умножение матриц: **A*B**	При умножении матриц или матриц и векторов число столбцов первого сомножителя должно быть равно числу строк второго. Если один из сомножителей – скаляр, то он умножается на все элементы другого сомножителя
.*	Умножение массивов: **A.*B**	Поэлементное перемножение двух массивов одинакового размера. На скаляр умножаются все элементы сомножителя
inv(A)	Функция обращения матрицы *A*	Вычисляется матрица, обратная заданной *A*. Исходная матрица *A* должна быть квадратной и невырожденной (**det(A)≠0**)
\	Решение систем линейных уравнений: **AX=B**	*A* – квадратная невырожденная матрица размера $n \times n$, *B* – прямоугольная матрица размера $n \times k$; решение по методу исключения Гаусса имеет вид: **X=A\B**. Операция **A\B** равносильна операции произведению матриц: **inv(A)*B**. Случай прямоугольной матрицы *A* размера $m \times n$ см. в [2, 5]
.\	Левое деление массивов: **A.\B**	Результатом является матрица с элементами **B(i,j)/A(i,j)**; матрицы должны быть одинаковых размеров за исключением случая, когда одна из них вырождается в скаляр

Таблица 1. Окончание

/	Решение систем линейных уравнений: **XA=B**	*A* – квадратная матрица размера *n*×*n*, *B* – прямоугольная матрица размера *k*×*n*; решение имеет вид: **X=B/A**. Операция **B/A** равносильна операции: **B*inv(A)**
./	Правое деление массивов: **A./B**	Результатом является матрица с элементами **A(i,j)/B(i,j)**; матрицы должны быть одинаковых размеров за исключением случая, когда одна из них вырождается в скаляр
^	Степень матрицы: **A^p**	Если *p* – целое положительное число, то степень матрицы вычисляется перемножением ее на себя *p* раз; если *p* – целое отрицательное число, то то же самое относится к обратной матрице. Для других значений *p* вычисление степени матрицы *A* связано с нахождением ее собственных значений
.^	Степень массива: **A.^B**	Результатом является матрица с элементами **A(i,j)^B(i,j)**; матрицы должны быть одинаковых размеров за исключением случая, когда одна из них вырождается в скаляр
'	Транспонирование матрицы: A'	Например, *A*' – транспонированная матрица *A*; для комплексных матриц транспонирование дополняется сопряжением
.'	Транспонирование массива: A.'	Для действительных и комплексных массивов строки просто заменяются столбцами; для комплексных массивов операция сопряжения не выполняется

Ниже приведен ряд примеров на действие с массивами и матрицами.

Операции с массивами. Обработка массивов выполняется поэлементно. Это относится не только к арифметическим или логическим операциям над массивами (одинакового размера), но и применению к ним всех указанных ранее (п. 2.4) элементарных математических функций. Любая из этих функций формирует матрицу того же размера, что и заданная матрица. Примеры:

```
>> A = [0 1 2 3 4; 6 -3 1 7 -2]
A =
     0    1    2    3    4
     6   -3    1    7   -2


>> B = sin(A)  % Функция sin берется от каждого элемента массива A
```

```
B =
    0         0.8415    0.9093    0.1411   -0.7568
   -0.2794   -0.1411    0.8415    0.6570   -0.9093

>> A+B  % Результат не зависит от перестановки операндов A и B
ans =
          0    1.8415    2.9093    3.1411    3.2432
      5.7206   -3.1411    1.8415    7.6570   -2.9093

>> A-2  % Тот же результат при: -2+A
ans =
     -2   -1    0    1    2
      4   -5   -1    5   -4

>> A./(A-2)  % Правое деление - равносильно левому делению: (A-2).\A
Warning: Divide by zero.
ans =
          0   -1.0000       Inf    3.0000    2.0000
      1.5000    0.6000   -1.0000    1.4000    0.5000

>> A.\(A-2)  % Левое деление - равносильно правому: (A-2)./A
Warning: Divide by zero.
ans =
        -Inf   -1.0000         0    0.3333    0.5000
      0.6667    1.6667   -1.0000    0.7143    2.0000

>> A./10  % Каждый элемент массива A делится на скаляр 10
ans =
          0    0.1000    0.2000    0.3000    0.4000
      0.6000   -0.3000    0.1000    0.7000   -0.2000

>> 4./A  % Скаляр 4 делится на каждый элемент массива A
Warning: Divide by zero.
ans =
         Inf    4.0000    2.0000    1.3333    1.0000
      0.6667   -1.3333    4.0000    0.5714   -2.0000

>> A.*B  % Сомножители A и B перестановочны: A.*B = B.*A
ans =
          0    0.8415    1.8186    0.4234   -3.0272
     -1.6765    0.4234    0.8415    4.5989    1.8186

>> A.*2  % Тот же результат при умножении: 2.*A, 2*A, A*2
ans =
       0    2    4    6    8
      12   -6    2   14   -4



% Поэлементное возведение в степень: C(i, j) = A(i, j)^B(i, j)
```

```
>> C = A.^B
C =
   1.0000   1.0000           1.8781   1.1677    0.3502
   0.6061   0.7736-0.3674i   1.0000   3.5910   -0.5110-0.1497i
```

Замечание: В результате произведенных вычислений массив **C(i, j)** содержит комплексные элементы. Это возникает в тех случаях, когда в выражении **A(i,j)^B(i,j)** основание **A(i,j)** – отрицательно, а показатель степени **B(i,j)** является дробным числом.

```
>> B.^A  % Поэлементное возведение в степень: B(i, j)^A(i, j)
ans =
      1.0000      0.8415    0.8268    0.0028    0.3280
      0.0005   -355.8232    0.8415    0.0528    1.2095

>> A.^2  % Каждый элемент массива A возведен в квадрат
ans =
      0     1     4     9    16
     36     9     1    49     4

% Возведение в квадрат обратной величины каждого элемента A
>> A.^(-2)
ans =
        Inf    1.0000    0.2500    0.1111    0.0625
     0.0278    0.1111    1.0000    0.0204    0.2500
```

```
>> C'
ans =
      1.0000    0.6061
      1.0000    0.7736+0.3674i
      1.8781    1.0000
      1.1677    3.5910
      0.3502   -0.5110+0.1497i
```

```
>> C.'
ans =
      1.0000    0.6061
      1.0000    0.7736-0.3674i
      1.8781    1.0000
      1.1677    3.5910
      0.3502   -0.5110-0.1497i
```

Замечание: Действие **C'** означает транспонирование матрицы *C*, а **C.'** – массива. В первом случае одновременно с транспонированием (строки поменялись местами со столбцами) произошла смена знаков перед мнимыми элементами комплексного массива **C** (операция сопряжения). Во втором случае операция комплексного сопряжения не применяется. Для действительных массивов операции, связанные с применением знаков: «**'**» и «**.'**», являются эквивалентными.

Операции с матрицами. К матричным действиям с матрицами относятся такие операции, которые используются в матричном исчислении в математике. Введем следующие матрицы:

```
>> A = [1:5; 6:9 11]
```

```
>> B = [4 4 5 7 8; 0:-1:-4]
```

```
A =                                   B =
   1    2    3    4    5                4    4    5    7    8
   6    7    8    9   11                0   -1   -2   -3   -4

>> A+B  % Операции сложения и вычитания такие же, как у массивов
ans =
     5     6     8    11    13
     6     6     6     6     7

>> A*2  % Умножение на скаляр; результат такой же, как при A.*2
ans =
     2     4     6     8    10
    12    14    16    18    22

>> A'  % Пример транспонирования матрицы
ans =
     1     6
     2     7
     3     8
     4     9
     5    11

>> A'*B  % Пример умножения матрицы на матрицу
ans =
     4    -2    -7   -11   -16
     8     1    -4    -7   -12
    12     4    -1    -3    -8
    16     7     2     1    -4
    20     9     3     2    -4

>> C = A*B'  % Пример умножения матрицы на матрицу
C =
    95   -40
   243   -94

>> inv(C)  % Обращение матрицы (возможно только при det(C) ≠ 0)
ans =
    -0.1190    0.0506
    -0.3076    0.1203

>> disp(det(C))  % Вычисление определителя матрицы C
   790

>> inv(C)*C  % Проверка обращения: результат – единичная матрица
ans =
    1.0000    0.0000
    0.0000    1.0000
```

Замечание: Матричные функции **inv, det** (и многие другие, см. раздел 5: мат-

ричные функции) работают только с квадратными матрицами. Проверку правильности обращения матрицы можно осуществить и другим способом: инвертированием уже обращенной матрицы; результатом проверки является исходная матрица: **inv(inv(C)) = C**.

```
>> A = [3 2 1; -1 4 0; 1 -2 6]
A =
    3    2    1
   -1    4    0
    1   -2    6

>> X = A\B
X =
   -0.1951
    0.4512
    0.6829
```

```
>> B = [1:3]'
B =
   1
   2
   3

>> inv(A)*B
ans =
     -0.1951
      0.4512
      0.6829
```

Последние два примера демонстрируют решение системы уравнений. Процедура **X = A\B** обеспечивает решение системы **AX = B** по методу Гаусса и является предпочтительнее операции: **inv(A)*B**, использующей обращение матрицы.

```
>> C = A ^2
C =
    8    12     9
   -7    14    -1
   11   -18    37
```

```
>> A ^(-2)
ans =
      0.0744   -0.0901   -0.0205
      0.0369    0.0293   -0.0082
     -0.0042    0.0410    0.0291
```

```
>> inv(ans)  % Проверка правильности возведения матрицы в степень
ans =
      8    12     9
     -7    14    -1
     11   -18    37
```

Замечание: При показателе степени p = -2 операция возведения матрицы A в степень сводится к возведению в квадрат обратной матрицы – A^{-1}. Последний пример иллюстрирует проверку, согласно которой результат инвертирования предыдущего значения переменной ans, равной ans = $(A^{-1})*(A^{-1})$, совпадает со значением матрицы $C = A^2$. Действительно, по правилу обращения матриц имеем [6]: $\text{inv}((A^{-1})*(A^{-1})) = (A^{-1}*A^{-1})^{-1} = (A^{-1})^{-1}*(A^{-1})^{-1} = A*A = C$. Более сложные случаи матричных операций по возведению в степень, когда в выражении A^p (**A^p**) число p – не целое или когда нужно вычислить «число в степени матрица»: p^A (**p^A**), связаны со спектральным разложением квадратной матрицы A.

4. ОБРАБОТКА ДАННЫХ В МАССИВАХ

4.1. Перечень основных функций

В MATLAB описаны функции, которые предназначены для анализа и обработки данных, заданных в виде числовых массивов. Ниже рассмотрены простейшие функции для обработки данных:

- суммирование элементов массива (**sum, cumsum**);
- произведение элементов массива (**prod, cumprod**);
- нахождение максимального и минимального элементов массива (**max, min**);
- нахождение средних, срединных значений массива и стандартных отклонений (**mean, median, std**);
- сортировка элементов массива (**sort, sortrows, cplxpair**);
- определение ковариационной матрицы элементов массива (**cov**);
- определение коэффициентов корреляции элементов массива (**corrcoef**);
- вычисление конечных разностей и приближенное дифференцирование, вычисление градиента функции от 2-ух переменных (**diff, gradient**);
- пятиточечная аппроксимация Лапласиана (**del2**).

Характерной чертой применения функций **sum, prod, max, min, mean, sort, diff** и т. д. является то, что действие данных функций распространяется не на строки массива (как это делается в векторах), а на каждый из столбцов заданного массива, т. е. столбец массива **A** рассматривается как переменная, а каждая строка – как наблюдение. Так, в результате применения функций **max, min, mean, std** получаются вектор-строки с числом элементов, равным числу столбцов заданного массива. Каждый элемент содержит, соответственно, максимальное, минимальное, среднее или среднеквадратичное значение элементов соответствующего столбца заданного массива.

4.2. Суммирование и произведение элементов массива

Функция **s = sum(A)** в случае одномерного массива возвращает сумму элементов массива; в случае двумерного массива – это вектор-строка, содержащая суммы элементов каждого столбца. Функция **s = cumsum(A),** кроме того, возвращает все промежуточные результаты суммирования. Для примера рассмотрим массив:

```
A =
     3     2     1
    -1     4     0
     1    -2     6

>> sum(A)
ans =
     3     4     7
```

```
>> cumsum(A)
ans =
     3     2     1
     2     6     1
     3     4     7
```

В последних версиях MATLAB предусмотрено расширение списка параметров функции **sum**: **sum(A, p)**, где p = 1 или 2. При **sum(A, 2)** происходит суммирова-

ние по строкам. В частности,

```
% Операция эквивалентна обращению к функции sum(A.')
>> sum(A, 2)
ans =
     6
     3
     5
```

Заметим, что **sum(A)** и **sum(A, 1)** приводят одинаковым результатам.

Функция **p = prod(A)** вычисляет произведение элементов массива и работает аналогично функции **sum(A)**:

```
>> prod(A)
ans =
     -3  -16    0
```

```
>> cumprod(A)
ans =
      3    2    1
     -3    6    0
     -3  -16    0
```

```
>> prod(A, 2)
ans =
      6
      0
    -12
```

4.3. Нахождение максимального и минимального элементов массива

Функции **max(A), min(A)** вычисляют вектор-строку, содержащие максимальные и минимальные элементы в соответствующих столбцах массива:

```
>> max(A)
ans =
     3    4    6
```

```
>> min(A)
ans =
     -1   -2    0
```

Таким образом, **max(max(A))** (функция **min(min(A))**) представляет собой наибольший (наименьший) элемент массива.

Функция **max(A, B)** (функция **min(A, B)**) возвращает массив того же размера, что **A** и **B,** каждый элемент которого есть максимальный (минимальный) из соответствующих элементов этих массивов. Функция **[Y, I] = max(A, B)** (функция **[Y, I] = min(A, B)**) кроме самих максимальных (минимальных) элементов возвращает вектор-строку индексов **I** этих элементов в данном столбце:

```
% Значения индексов отвечают номерам строк в A
>> [Y, I] = max(A)
Y =
   3    4    6
I =
   1    2    3
```

4.4. Нахождение средних, срединных значений массива и стандартных отклонений

Элементарная статистическая обработка данных в массиве обычно сводится к нахождению их среднего значения, медианы (срединного значения) и стандартного отклонения. Для этой цели в системе MATLAB предусмотрены следующие функции: **mean, median, std**.

Функция **mean(A)** возвращает арифметическое среднее элементов массива, если *A* – вектор или – вектор-строку, содержащую арифметическое среднее элементов каждого столбца, если *A* – матрица. Арифметическое среднее это сумма элементов массива, деленное на их число. Таким образом, **mean(mean(A))** это арифметическое среднее (математическое ожидание) элементов массива.

mean(A, p) возвращает среднее значение элементов массива по столбцам (при p =1) и по строкам (при p = 2).

Функция **median(A)** возвращает значение срединного элемента (медиану), если *A* – вектор; или вектор-строку медиан для каждого столбца, если *A* – матрица. Функция **median(median(A))** это срединный элемент (медиана) массива, что совпадает со значением **median(A(:))**.

median(A, p) возвращает значения медиан для столбцов (p =1) и для строк (p =2).

Функция **std(A)** возвращает стандартное отклонение элементов массива, вычисляемое по формуле:

$$S = \left[\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right]^{\frac{1}{2}},$$

где: *A* – вектор. Здесь $\bar{x}$ = **mean(A)**; x_i – элемент вектора *A*. Если *A* – матрица, **std(A)** возвращает вектор-строку, содержащую стандартное отклонение элементов каждого столбца.

Функция **std(A, flag)** вычисляет то же значение, что и **std(A),** если flag = 0; если flag = 1, **std(A, 1)** вычисляет стандартное отклонение по формуле:

$$S = \left[\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right]^{\frac{1}{2}}.$$

Функция **std(A, flag, p)** возвращает стандартное отклонение по строкам или столбцам для матрицы *A* в зависимости от значения переменной p.

Примеры:

```
>> A = [1 4 2 6 9; 10 7 3 5 8; 5 5 2 1 6]
A =
    1    4    2    6    9
   10    7    3    5    8
    5    5    2    1    6

>> mean(A)  % Определение средних значений элементов массива A
ans =
    5.3333    5.3333    2.3333    4.0000    7.6667

>> mean(A, 2)  % Средние значения элементов вычисляются по строкам
ans =
```

```
     4.4000
     6.6000
     3.8000

>> median(A)  % Определение срединных значений (медиан)
ans =
     5    5    2    5    8

>> median(A, 2)  % Значения медиан вычисляются по строкам
ans =
     4
     7
     5

>> std(A)  % Определение стандартного отклонения элементов массива
ans =
     4.5092    1.5275    0.5774    2.6458    1.5275

>> std(A, 2)  % То же – вычисляется по строкам
ans =
     3.2094
     2.7019
     2.1679
```

4.5. Сортировка элементов массива

Многие операции статистической обработки данных выполняются быстрее и надежнее, если данные предварительно отсортированы. Этой цели служит ряд функций MATLAB.

Функция **sort(A)** упорядочивает элементы одномерного массива по возрастанию; для двумерного массива упорядочиваются элементы каждого столбца.

Функция **[B, I] = sort(A)** кроме массива упорядоченных элементов по столбцам (**B**) возвращает массив индексов **I** (по размеру **A**), позволяющих восстановить структуру исходного массива. Функция **sort(A, p)** для двумерных массивов сортирует элементы по строкам или столбцам в зависимости от значения переменной *p*.

sortrows(A) – выполняет сортировку строк массива **A** по возрастанию и возвращает отсортированный массив, который может быть или матрицей, или вектором-столбцом.

sortrows(A, X) – возвращает матрицу, отсортированную по столбцам, точно указанным в векторе *X*. Например, **sortrows(A, [2 3])** сортирует строки матрицы *A* сначала по второму столбцу, который может быть или матрицей, или вектором-столбцом.

Функция **cplxpair(A)** – сортирует элементы комплексного массива **A**. Для массива **A = [1 4 2 6 9; 10 7 3 5 8; 5 5 2 1 6]** рассмотрены примеры:

```
>> sort(A) % Сортировка элементов массива A по возрастанию
ans =
      1    4    2    1    6
      5    5    2    5    8
     10    7    3    6    9

>> [B, I] = sort(A)  % B – отсортированный массив, I – массив индексов
B =
    1    4    2    1    6
    5    5    2    5    8
   10    7    3    6    9
I =
    1    1    1    3    3
    3    3    3    2    2
    2    2    2    1    1

>> B = sortrows(A)
B =
    1    4    2    6    9
    5    5    2    1    6
   10    7    3    5    8

>> B = sortrows(A, 4)
B =
    5    5    2    1    6
   10    7    3    5    8
    1    4    2    6    9
```

4.6. Определение матрицы ковариаций и коэффициентов корреляции элементов массива

Рассмотрены функции для вычисления ковариационной матрицы и коэффициентов корреляции.

Функция **C = cov(A)** в случае одномерного массива возвращает дисперсию элементов массива; в случае двумерного массива, когда каждый столбец рассматривается как переменная, а каждая строка – как наблюдение, функция **cov(A)** вычисляет матрицу ковариаций, функция **diag(cov(A))** – вектор дисперсий, а **sqrt(diag(cov(A)))** – вектор стандартных отклонений для каждого столбца.

Вычисление матрицы ковариаций реализуется следующим алгоритмом:

n = size(A, 1); B = A – ones(n, 1)*mean(A); C = B'*B/(n – 1);

Функция **S = corrcoef(A)** возвращает матрицу коэффициентов корреляции для двумерного массива (каждый столбец – переменная, каждая строка – наблюдение). Элементы матрицы **S = corrcoef(A)** связаны с элементами матрицы ковариаций **C = cov(A)** следующим соотношением:

$$S(i,j) = \frac{C(i,j)}{\sqrt{C(i,i)C(j,j)}}.$$

Примеры:

```
>>C = cov(A)  % Вычисление матрицы ковариации
C =
   20.3333    6.8333    2.3333   -1.5000   -1.8333
    6.8333    2.3333    0.8333         0   -0.3333
    2.3333    0.8333    0.3333    0.5000    0.1667
   -1.5000         0    0.5000    7.0000    4.0000
   -1.8333   -0.3333    0.1667    4.0000    2.3333

>>S = corrcoef(A)  % Вычисление коэффициентов корреляции
S =
    1.0000    0.9921    0.8963   -0.1257   -0.2662
    0.9921    1.0000    0.9449         0   -0.1429
    0.8963    0.9449    1.0000    0.3273    0.1890
   -0.1257         0    0.3273    1.0000    0.9897
   -0.2662   -0.1429    0.1890    0.9897    1.0000
```

4.7. Вычисление конечных разностей и приближенное дифференцирование, приближенное вычисление градиента функции от двух переменных

Одним из важнейших приложений конечно-разностных методов является приближенное представление производных функций – численное дифференцирование. Оно реализуется функцией **diff**.

Функция **diff(A)** возвращает конечные разности смежных элементов массива *A*. Если *A* – вектор, то **diff(A)** возвращает вектор разностей соседних элементов **[A(2) -A(1) A(3) -A(2) … A(n) -A(n-1)]**, у которого количество элементов на единицу меньше, чем у исходного вектора *A*. Если *A* – матрица, то **diff(A)** возвращает матрицу разностей столбцов: **[A(2:m, :) -A(1:m-1, :)]**.

Функция **y = diff(A, n)** вычисляет конечные разности порядка *n*, удовлетворяющие рекуррентному соотношению: **diff(A, n) = diff(A, n-1)**. Так, **diff(A, 2)** возвращает конечные разности 2-го порядка. Эту же операцию можно получить по команде: **diff(diff(A))**.

diff(A, n, p) вычисляет конечные разности по строкам или по столбцам для массива *A* в зависимости от параметра *p*. Если порядок числа *n* равен или превышает величину *p*, то **diff** возвращает пустой массив.

Вычисление конечно-разностным методом градиента функций реализуется с помощью функции **FX = gradient(F)**. Эта функция возвращает одномерный градиент от вектора *F*. Результат *FX* соответствует конечным разностям в направлении *x*.

[FX, FY] = gradient(F) вычисляет двумерный градиент от матрицы *F* в виде массивов *FX, FY*. *FX* соответствует конечным разностям в направлении *x* (столбцов). *FY* соответствует конечным разностям в направлении *y* (строк).

[FX, FY, FZ,…] = gradient(F) возвращает *N* компонентов градиента от многомерного массива *F*.

[…] = gradient(F, h) использует шаг *h* для установки расстояния между точка-

ми в каждом направлении (h – скалярная величина; по умолчанию $h = 1$).

[…] = gradient(F, h1, h2, …) – если **F** многомерный массив, то расстояния задаются с помощью шагов h_1, h_2,….

Примеры:

```
>> X = [1 2 4 6 7 9 3 45 6 7]
X =
   1    2    4    6    7    9    3   45    6    7

>> size(X)  % Определение размеров вектора X
ans =
     1   10

>> Y = diff(X)
Y =
   1    2    2    1    2   -6   42  -39    1

>> size(Y)  % После дифференцирования длина Y уменьшилась на 1
ans =
       1    9

% Повторное дифференцирование вектора X: diff(diff(X))
>> Y = diff(X, 2)
Y =
   1    0   -1    1   -8   48  -81   40

% Массив A = [1 4 2 6 9; 10 7 3 5 8; 5 5 2 1 6] введен в п. 4.4
>> Y = diff(A)
Y =
    9    3    1   -1   -1
   -5   -2   -1   -4   -2

% Повторное дифференцирование массива: diff(diff(A))
>> Y = diff(A, 2)
Y =
   -14   -5   -2   -3   -1

% Система сообщает, что Y - пустой массив (p > n!)
>> Y = diff(A, 2, 3)
Y =
    Empty array: 3-by-5-by-0

% Градиент для вектора X = [1 2 4 6 7 9 3 45 6 7]
>> FX = gradient(X)
FX =
 Columns 1 through 7
    1.0000   1.5000   2.0000   1.5000   1.5000  -2.0000  18.0000
  Columns 8 through 10
    1.5000 -19.0000   1.0000

% Двумерный градиент: массивы чисел FX, FY
>> [FX, FY] = gradient(A)
```

```
FX =
    3.0000    0.5000    1.0000    3.5000    3.0000
   -3.0000   -3.5000   -1.0000    2.5000    3.0000
         0   -1.5000   -2.0000    2.0000    5.0000
FY =
    9.0000    3.0000    1.0000   -1.0000   -1.0000
    2.0000    0.5000         0   -2.5000   -1.5000
   -5.0000   -2.0000   -1.0000   -4.0000   -2.0000
```

4.8. Пятиточечная аппроксимация Лапласиана

Для выполнения аппроксимации Лапласиана в MATLAB используется функция **del2**.

Функция **L = del2(U)** возвращает массив *L* того же размера, что и массив *U*. При этом каждый элемент массива *L* равен разности среднего значения соседних элементов и элемента рассматриваемого узла для массива *U*. Узлы сетки во внутренней области имеют четырех «соседей», а на границе и в углах – только трех или двух «соседей».

Если массив *U* рассматривать как функцию двух переменных *U*(*x*, *y*), вычисленную в точках квадратной сетки, то процедура **4*del2(U)** является конечно-разностной аппроксимацией дифференциального оператора Лапласа, примененного к функции *U*:

$$\Delta^2 \boldsymbol{u} = \frac{\partial^2 \boldsymbol{u}}{\partial^2 \boldsymbol{x}} + \frac{\partial^2 \boldsymbol{u}}{\partial^2 \boldsymbol{y}}.$$

Другие функции также вычисляют массив *L*, но допускают дополнительные установки.

Функция **L = del2(U, h)** использует шаг *h* для установки расстояния между точками в каждом координатном направлении; по умолчанию *h* = 1 (*h* – скалярная величина).

L = del2(U, hx, hy) использует шаг *hx* и *hy* для точного определения расстояния между точками. Если *hx* (*hy*) – скалярная величина, то задается расстояние между точками в направлении оси *x* (оси *y*), если *hx* (*hy*) – вектор, то он должен иметь длину, равную **size(U, 2)** (для *hy* – **size(U, 1)**), и точно определять координаты точек по оси *x* (оси *y*).

L = del2(U, hx, hy, hz,…) применяется для случая многомерного массива **U**; расстояния задаются с помощью шагов *hx*, *hy*, *hz*,….

В качестве примера рассмотрим функцию

$$U(x, y) = x^2 + y^2,$$

имеющую лапласиан: $\Delta^2 u = 4$. Это подтверждают графики данных функций (см. рис. 30).

```
% Создание сетки на плоскости x-y
>> [x, y] = meshgrid(-4:4, -3:3);
```

```
>> U = x.*x + y.*y
U =
     25   18   13   10    9   10   13   18   25
     20   13    8    5    4    5    8   13   20
     17   10    5    2    1    2    5   10   17
     16    9    4    1    0    1    4    9   16
     17   10    5    2    1    2    5   10   17
     20   13    8    5    4    5    8   13   20
     25   18   13   10    9   10   13   18   25

>> L = 4*del2(U)  % 5-точечная формула аппроксимации Лапласиана
L =
      4    4    4    4    4    4    4    4    4
      4    4    4    4    4    4    4    4    4
      4    4    4    4    4    4    4    4    4
      4    4    4    4    4    4    4    4    4
      4    4    4    4    4    4    4    4    4
      4    4    4    4    4    4    4    4    4
      4    4    4    4    4    4    4    4    4

% Создание графического окна с двумя подокнами (рис. 30, а)
>>subplot(1, 2, 1)
% Вывод сетчатой поверхности для значений массива U
>>surf(U)
>>subplot(1, 2, 2)  % Рис. 30, б
>>surf(L)
```

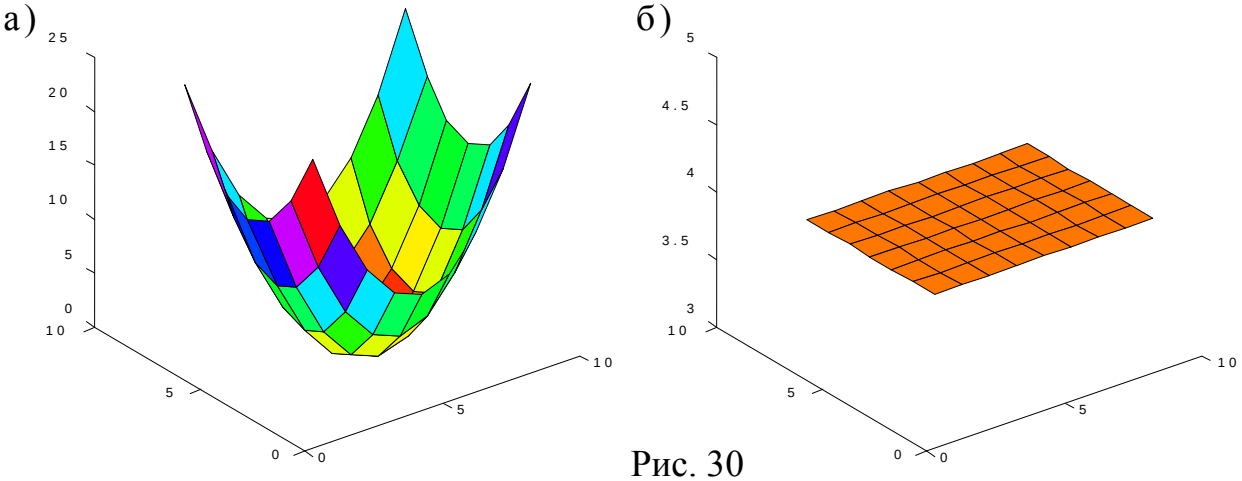

Рис. 30

Замечание: Для большей выразительности решаемой задачи по реализации функции **del2** в примере задействованы графические команды и функции **meshgrid, subplot, surf**, не рассматриваемые в настоящем пособии. Данные команды принадлежат к средствам графической визуализации и широко используются как в командном режиме вычислений, так и в программах. С работой графических функций можно познакомиться, например, в библиографии [1–3, 5, 7] либо в справочной системе MATLAB, стандартный вариант которой реализован в интерактивном командном режиме (команда **help**).

5. МАТРИЧНЫЕ ФУНКЦИИ

5.1. Матричные функции: expm, logm, sqrtm и funm

Система MATLAB это мощный инструмент численного анализа инженерных и научных задач [1–5], решение которых нередко требует вычисления функций от матриц.

В MATLAB выражения типа **exp(A)** и **sqrt(A)** рассматриваются как операции над отдельными элементами массива, определенного матрицей *A*. Однако в MATLAB предусмотрена возможность вычисления матричных функций, таких, как экспонента, логарифм и другие. Эти функции определены только для квадратных матриц.

Вычисление матричной экспоненты (e^A) производится с помощью встроенной функции **expm** (см. также функции **expm1**, **expm2** и **expm3** [1–3, 5]). Рассмотрим примеры:

```
>> A = [1 -1 -1; 1 0 4; -1 2 1]
A =
    1   -1   -1
    1    0    4
   -1    2    1

>>expm(A)  % Вычисление матричной экспоненты
ans =
     4.3058   -9.1923  -14.8171
    -2.0573   11.9880   19.8947
    -3.5675   12.7598   21.1803

>>exp(A)  % Вычисляется экспонента от каждого элемента массива A
ans =
     2.7183    0.3679    0.3679
     2.7183    1.0000   54.5982
     0.3679    7.3891    2.7183
```

Вычисление логарифма от матрицы производится с помощью встроенной функции **logm(A),** которая вычисляет функцию, обратную **expm(A)**. Результат получается комплексным, если *A* имеет отрицательные собственные значения. Пример:

```
>> logm(A)  % Вычисление матричного логарифма
ans =
    -0.4053-0.1794i   -0.3742+0.2308i   -0.9074-0.3441i
    -0.6923-1.3804i    0.8203+1.7759i   -0.1031-2.6476i
     0.1591+0.8056i    0.2151-1.0364i    1.1944+1.5451i

>> expm(logm(A))  % Проверка: expm(logm(A)) = logm(expm(A)) = A
ans =
     1.0000-0.0000i   -1.0000+0.0000i   -1.0000-0.0000i
     1.0000+0.0000i    0.0000-0.0000i    4.0000-0.0000i
    -1.0000+0.0000i    2.0000-0.0000i    1.0000+0.0000i
```

Вычисление функции $A^{1/2}$ производится с помощью встроенной функции **sqrtm**. Функция **sqrtm(A)** вычисляет одну из многих матриц $B = A^{1/2}$, которые удовлетворяют условию: $B*B = A$. Пример:

```
>>sqrtm(A)  % Извлечение корня квадратного из матрицы A
ans =
      0.9180-0.0848i   -0.3532+0.1091i   -0.4007-0.1626i
      0.2580-0.6524i    0.7076+0.8393i    1.2699-1.2513i
     -0.3056+0.3807i    0.6587-0.4898i    1.0607+0.7302i

>>ans*ans  % Проверка: полученный результат совпадает с исходной A
ans =
      1.0000-0.0000i   -1.0000+0.0000i   -1.0000-0.0000i
      1.0000+0.0000i    0.0000-0.0000i    4.0000-0.0000i
     -1.0000+0.0000i    2.0000-0.0000i    1.0000+0.0000i
```

Для сравнения:

```
>>sqrt(A)  % Извлечение корня квадратного из каждого элемента A
ans =
      1.0000     0+1.0000i     0+1.0000i
      1.0000             0        2.0000
      0+1.0000i     1.4142        1.0000
```

Заметим, что если этот результат возвести в квадрат (по правилу **ans^2** или **ans*ans**), то полученная матрица не совпадет по своему значению с матрицей *A*.

Вычисление произвольных функций от матрицы. Функция

Y = funm(A, '<имя функции>')

позволяет вычислить любую функцию от матрицы, если она имеет имя, составленное из латинских букв. В частности, это могут быть все элементарные математические функции (см. п. 2.4). Команды **funm(A, 'sqrt')** и **funm(A, 'log')** эквивалентны командам **sqrtm(A)** и **logm(A)**. Команды **funm(A, 'exp')** и **expm(A)** вычисляют одинаковую матричную функцию, но используют разные алгоритмы. При этом команда **expm(A)** является более предпочтительной. Примеры:

```
>>Y = funm(A, 'exp')  % Сравните с результатом вычисления по expm(A)
Y =
    4.3058     -9.1923    -14.8171
   -2.0573     11.9880     19.8947
   -3.5675     12.7598     21.1803

>> funm(A, 'sin')  % Вычисление синуса от матрицы A
ans =
      0.6040      0.1691      0.6038
      0.7001     -0.5308      0.6274
     -0.2655      0.0964     -0.6999
```

5.2. Матричные функции линейной алгебры

Матричные функции позволяют решать сложные задачи в различных областях науки и техники. В MATLAB реализованы следующие группы функций, предназначенные для:

- вычисления чисел обусловленности матрицы (**cond, condeig, rcond**);
- вычисления определителя и ранга матрицы (**det, rank**);
- определения векторной и матричной нормы (**norm**);
- определения ортонормального базиса матрицы (**orth, null**);
- обращения матриц и вычисления следа матрицы (**inv, pinv, trace**);
- вычисления собственных значений и сингулярных чисел матрицы (**eig, svd**);
- приведения матрицы к треугольной форме (**rref, rrefmovie**);
- разложения Холецкого (**chol, cholinc**);
- *LU*- и *QR*-разложения (**lu, luinc, qr**);
- приведения матриц к форме Шура и Хессенберга (**schur, rsf2csf, hess, qz**).

Ниже приведены основные матричные функции каждой группы и даны их краткие характеристики.

5.3. Скалярные характеристики матриц

К данной категории относятся матричные функции первых трех групп: функции вычисления чисел обусловленности, определителя и ранга матрицы, а также функции определения векторной и матричной нормы.

Вычисления чисел обусловленности матрицы

Числа обусловленности матрицы определяют чувствительность решения системы линейных уравнений к погрешностям исходных данных.

Функция **k = cond(A)** вычисляет число обусловленности матрицы *A*, основанное на второй норме и равное отношению максимального сингулярного числа матрицы к минимальному. Значения **cond(A)** и **cond(A, p),** близкие к 1, указывают на хорошо обусловленную матрицу.

k = cond(A, p) – возвращает число обусловленности матрицы, основанное на *p*-норме: **norm(A, p)*norm(inv(A), p)** (о *p*-норме см. функцию **norm**). Пример:

```
>> H = hilb(4)  % Формирование матрицы Гильберта 4-го порядка
H =
   1.0000    0.5000    0.3333    0.2500
   0.5000    0.3333    0.2500    0.2000
   0.3333    0.2500    0.2000    0.1667
```

```
   0.2500    0.2000    0.1667    0.1429

>> k = cond(H)  % Число обусловленности матрицы Гильберта
k =
   1.5514e+004
```

Функция **c = rcond(A)** вычисляет величину, обратную значению числа обусловленности матрицы *A* относительно 1-нормы. Если *A* хорошо обусловленная, то значение **c** близко к 1.00, если плохо обусловленная – то около 0.

Пример:

```
>> C = rcond(hilb(4))  % Результат соответствует C = 1/cond(A, 1)
C =
   4.6461e-005
```

Вычисления определителя и ранга матрицы

Функция **d = det(A)** вычисляет определитель квадратной матрицы; если матрица *A* целочисленная, то результатом является также целое число. Определитель матрицы вычисляется на основе треугольного разложения методом исключения Гаусса.

Функция **r = rank(A)** возвращает число сингулярных значений, которые являются большими, чем заданный по умолчанию допуск. **rank(A, t)** возвращает число сингулярных значений, которые являются большими, чем **t**.

Примеры:

```
>>det(A)  % Определитель матрицы A
ans =
     -5

>>rank(A)  % Ранг матрицы A
ans =
     3
```

Определение векторной и матричной нормы

Функция **k = norm(V, p)** вычисляет *p*-норму вектора *V* по формуле:

$$\mathbf{k = sum(abs(V).^p)^(1/p)},$$

где параметр *p* – целое положительное число. Если аргумент *p* при обращении к функции не указан, вычисляется 2-ая норма.

Функция **k = norm(A, p)** вычисляет *p*-норму матрицы *A* по формуле:

$$\mathbf{k = max(sum(abs(A).^p))^(1/p)},$$

где *p* = 1, 2, **inf** или **'fro'**; **'fro'** – норма Фробениуса (Frobenius). Если аргумент *p* не указан, задается 2-ая норма. При этом справедливы соотношения:

$$\mathbf{norm(A, 1) = max(sum(abs(A)))},$$
$$\mathbf{norm(A) = norm(A, 2)} = \sigma_{max}(\mathbf{A}),$$

norm(A, inf) = max(sum(abs(A'))),
norm(A, 'fro') = sqrt(sum(diag(A'*A))).

Здесь $\sigma_{max}(\mathbf{A})$ – наибольшее сингулярное число матрицы *A* [6].

Примеры вычисления различных норм вектора *X* и матрицы *A*:

```
>> X = [1 2 3];

>> disp(norm(X))
   3.7417

>> disp(norm(X, 1))
   6

>> disp(norm(X, 2))
   3.7417

>> disp(norm(X, inf))
   3
```

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

>> disp(norm(A))
   16.8481

>> disp(norm(A, 1))
   18

>> disp(norm(A, 2))
   16.8481

>> disp(norm(A, inf))
   24

>> disp(norm(A, 'fro'))
   16.8819
```

5.4. Матричные функции: orth, null, inv, pinv, trace

В данном разделе рассмотрены функций, связанные с определением ортонормального базиса матрицы, обращением матрицы и вычислением его следа.

Определение ортонормального базиса матрицы

Функция **Q = orth(A)** возвращает ортонормальный базис матрицы *A*; столбцы *Q* определяют то же пространство, что и столбцы *A*, но столбцы *Q* ортогональны, т. е.: *Q*'**Q* = *E*, где *E* – единичная матрица. В MATLAB это эквивалентно действиям: **Q'*Q = eye(size(A))**. Количество столбцов матрицы *Q* определяет ранг *A*, что можно вычислить так: **rank = size(orth(A), 2)**.

Функция **Q = null(A)** возвращает ортонормальный базис нуль-пространства матрицы *A*; если *Q* – не пустая матрица, то справедливы следующие соотношения: *Q*'**Q* = *E*, *A***Q* = 0. Количество столбцов матрицы *Q* определяет размерность нуль-пространства или дефект матрицы *A*, что вычисляется по следующей операции: **defect = size(null(A), 2)**.

В качестве примера для матрицы **A** = [1 2 3;4 5 6;7 8 9] получим:

```
>> orth(A)
ans =
     0.2673    0.8729
     0.5345    0.2182
     0.8018   -0.4364
```

```
>> null(A)
ans =
      0.4082
     -0.8165
      0.4082
```

Обращение матриц и вычисления следа матрицы

Обращение матриц – одна из наиболее распространенных операций матричного анализа. Обратной называют матрицу A^{-1}, получаемую в результате деления единичной матрицы E на исходную A: **eye(size(A))/*A***.

Функция **inv(A)** возвращает матрицу, обратную квадратной матрице A. На практике вычисление явной обратной матрицы требуется не так часто. Как правило, говоря о задаче обращения, имеют в виду нахождение решений систем линейных уравнений: ***AX* = *B*, *XA* = *B***. В рамках MATLAB для этих целей рекомендуется использовать решатели систем, основанные на методе исключения Гаусса без явного формирования A^{-1}. Это операторы вида: **X = A\B** или **X = B/A**, вместо, например, операции **X = inv(A)*B**.

Замечание: В процессе выполнения функции **inv** на компьютерах с IEEE-арифметикой (если A плохо масштабирована или близка к вырожденной) возможно появление следующего предупреждающего сообщения:

```
Matrix is singular to working precision.

При заданной точности матрица вырожденна.
```

При этом формируется матрица, все элементы которой равны **inf**. На машинах без IEEE-арифметики, например, на VAX эта ситуация рассматривается как ошибка.

Если выполненное обращение ненадежно, система диагностирует

```
Warning: Matrix is close to singular or badly scaled.
Results my be inaccurate. RCOND = xxx.

Предупреждение: Матрица близка к вырожденной.
Результаты могут быть неточными. RCOND = xxx.
```

Функция **P = pinv(A)** вычисляет матрицу, псевдообратную матрице A и удовлетворяет следующим условиям: $A*P*A = A$, $P*A*P = P$.

Пример:

```
>> inv(hilb(4))  % Обращение матрицы Гильберта
ans =
  1.0e+003*
     0.0160   -0.1200    0.2400   -0.1400
    -0.1200    1.2000   -2.7000    1.6800
     0.2400   -2.7000    6.4800   -4.2000
    -0.1400    1.6800   -4.2000    2.8000
```

След матрицы A, так же как и ее определитель, – одно из инвариантных свойств подобия матрицы A. След матрицы A определяется как сумма ее диагональных элементов и вычисляется с помощью функции **t = trace(A)**, возвращающей след квадратной матрицы.

Алгоритм вычисления следа матрицы в языке MATLAB это однострочный M-

файл: **t = sum(diag(A))**. Так, для найденной в предыдущем примере обратной матрицы Гильберта след составляет:

```
% Системная переменная ans содержит величину inv(hilb(4))
>>trace(ans)
ans =
     1.0496e+004
```

5.5. Вычисление собственных значений и сингулярных чисел матрицы

Проблема собственных значений матрицы и принадлежащих им собственных векторов играет важную роль во многих областях математических и прикладных наук. Математически проблема состоит в следующем: найти все нетривиальные (ненулевые) решения системы уравнений, представленных в скалярном виде:

$$AV_i = V_i\lambda_i \ (\mathrm{i} = 1, 2, \ldots, n),$$

где A – квадратная матрица порядка n; V_i – собственный вектор матрицы A; λ_i – число, называемое собственным значением матрицы A.

Матричная формулировка приведенной системы записывается следующим образом:

$$AV = VD.$$

Здесь V называется матрицей собственных векторов, так как содержит в своих столбцах собственные векторы V_i. Матрица D называется диагональной (или канонической) формой матрицы A, так как на главной диагонали этой матрицы расположены собственные значения λ_i: $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$.

Применительно к строительной механике проблема собственных значений встречается, например, при изучении вопросов устойчивости и колебаний сооружений, причем собственные элементы (λ_i, V_i) имеют строгий физический смысл. Так, в задачах устойчивости сооружений собственными значениями являются *критические силы*, а собственными векторами – соответствующие этим силам *формы потери устойчивости*. В динамике сооружений, в частности, при анализе свободных (или собственных) колебаний собственные значения и векторы определяют соответственно *частоты и формы собственных колебаний системы*. В связи с этим большое значение приобретают средства для вычисления собственных значений матрицы. Ниже дана подборка этих средств, реализованных в системе MATLAB.

Функция **d = eig(A)** вычисляет собственные значения матрицы A, определяемые вектором d.

Функция **[V, D] = eig(A)** вычисляет собственные векторы и собственные значения матрицы A. Собственные векторы содержатся в столбцах преобразующей матрицы V; собственные значения расположены на главной диагонали матрицы D, представляющей каноническую форму матрицы A. Вычисленные матрицы удовлетворяют матричному соотношению: **A*V = V*D**.

Аналогично предыдущему функции **d = eig(A, B), [V, D] = eig(A, B)** вычисля-

ют обобщенные собственные значения для случая более сложного уравнения $AV = BVD$, где A и B – квадратные матрицы.

Команда **S = svd(A)** выполняет сингулярное разложение матрицы A. В векторе S содержатся сингулярные числа матрицы A [6].

Примеры на вычисление собственных и сингулярных чисел матрицы A:

```
>> A = [3 2 1; 2 4 -1; 1 -1 6];

>> [V,D] = eig(A)                                  >> d = eig(A)
V =                                                d =
   -0.6526    -0.0937    -0.7519                      5.5302
   -0.7364     0.3120     0.6003                      6.4292
   -0.1784    -0.9454     0.2726                      1.0407
D =
    5.5302          0          0
         0     6.4292          0
         0          0     1.0407

>> S = svd(A)  % Вычисление сингулярных чисел матрицы A
S =
    6.4292
    5.5302
    1.0407
```

5.6. Матричные функции, связанные со специальными формами и разложениями

Многие задачи линейной алгебры (например, решение матричных уравнений Ляпунова, Сильвестра, Риккати) основаны на вычислительных методах, связанных с построением специальных матричных форм и разложений. Реализация данных вычислительных алгоритмов в MATLAB обеспечивается соответствующими матричными функциями.

Приведение матрицы к треугольной форме

Функция **R = rref(A)** осуществляет приведение матрицы A к треугольной форме на основе метода исключения Гаусса с частным выбором ведущего элемента. Функция **rrefmovie(A)** показывает пошаговое исполнение процедуры приведения матрицы A к треугольной форме.

Разложение Холецкого

Функция **R = chol(A)** находит разложение Холецкого для действительных симметрических и комплексных эрмитовых матриц. Если A – положительно определенная матрица, то матрица R – верхняя треугольная и удовлетворяет соотношению: $R'*R = A$; для произвольной матрицы A применение процедуры **chol(A)** приводит к сообщению об ошибке. Функция **cholinc** используется для вычисления неполного разложения Холецкого. Примеры:

```
>> A = [3 2 1; 2 4 -1; 1 -1 6]          >> B = [0 1 1;-1 0 1;-1 -1 0]
A =                                     B =
```

```
    3     2     1                          0     1     1
    2     4    -1                         -1     0     1
    1    -1     6                         -1    -1     0

>> C = A+i*B  % Формирование эрмитовой матрицы C
C =
    3.0000            2.0000+1.0000i    1.0000+1.0000i
    2.0000-1.0000i    4.0000           -1.0000+1.0000i
    1.0000-1.0000i   -1.0000-1.0000i    6.0000

>> R = chol(A)  % Треугольное разложение действительной матрицы A
R =
    1.7321    1.1547    0.5774
         0    1.6330   -1.0206
         0         0    2.1506

>> RC = chol(C)  % Треугольное разложение эрмитовой матрицы C
RC =
     1.7321    1.1547+0.5774i    0.5774+0.5774i
          0            1.5275   -1.3093+0.4364i
          0                 0            1.8516

>> R'*R  % Проверка: произведение R'*R дает исходную матрицу A
ans =
      3.0000    2.0000    1.0000
      2.0000    4.0000   -1.0000
      1.0000   -1.0000    6.0000

>> RC'*RC  % Проверка: RC'*RC = C
ans =
      3.0000            2.0000+1.0000i    1.0000+1.0000i
      2.0000-1.0000i    4.0000           -1.0000+1.0000i
      1.0000-1.0000i   -1.0000-1.0000i            6.0000
```

Замечание: Эрмитовой $C = A + i*B$ называется такая комплексная матрица, у которой действительная часть $\mathrm{Re}(C) = A$ – симметрическая, а мнимая $\mathrm{Im}(C) = B$ – кососимметрическая матрицы. Положительно определенной называется матрица, у которой все главные миноры всех порядков – больше нуля. Следствием этого является то, что у положительно определенной матрицы все собственные значения – положительны. Подробнее см. в [6].

LU- и QR-разложения

Функция **[L, U] = lu(A)** находит *LU*-разложение для произвольной квадратной матрицы A в виде произведения нижней треугольной матрицы L (возможно с перестановками) и верхней треугольной матрицы U: $A = L*U$.

Функция **[Q, R] = qr(A)** находит *QR*-разложение для произвольной матрицы A в виде произведения унитарной матрицы Q и верхней треугольной матрицы R, так

что $A = Q*R$. Для унитарной матрицы справедливо условие: $Q'*Q = E$, которое в MATLAB имеет вид: **Q'*Q = eye(size(A))**.

Приведения матриц к форме Шура и Хессенберга

Функция **T = schur(A)** возвращает матрицу в форме Шура *T*. *Комплексная форма Шура* – это верхняя треугольная матрица с собственными значениями на диагонали; *действительная форма Шура* сохраняет на диагонали действительные собственные значения, а комплексные представляются в виде блоков 2×2, частично занимая нижнюю поддиагональ.

Функция **[U, T] = schur(A)** кроме матрицы Шура *T* возвращает также унитарную матрицу преобразований *U*, которая удовлетворяет условиям: $A = U*T*U'$, $U'*U = E$.

Функция **[U, T] = rsf2csf(U, T)** преобразовывает действительную квазитреугольную форму Шура в комплексную треугольную.

Функция **H = hess(A)** возвращает матрицу в верхней форме Хессенберга. Структура данной форма характеризуется тем, что ее элементы, расположенные ниже первой поддиагонали (т. е. диагонали, находящейся непосредственно под главной диагональю), равны нулю. Если исходная матрица *A* симметрическая или эрмитова, то матрица Хессенберга вырождается в трехдиагональную.

Функция **[P, H] = hess(A)** кроме матрицы в верхней форме Хессенберга *H* возвращает также унитарную матрицу преобразований *P*, которая удовлетворяет условиям: $A = P*H*P'$, $P'*P = E$.

В отдельных случаях матричные линейные уравнения содержат в левой части не одну, а две заданных матрицы (*A* и *B*), как, например, в уравнении Сильвестра: $AX + XB = C$. Возможны и более сложные структуры в записи уравнений. Один из путей решения таких уравнений связан с необходимостью одновременного приведения пары матриц к форме Шура. Эту операцию выполняет *QZ*-алгоритм, который в MATLAB реализован функцией **qz**.

Ниже приведены примеры построения форм Шура и Хессенберга для матрицы с действительными и комплексно-сопряженными собственными значениями:

```
>> A = [3 2 1 0; -1 1 2 -1; 1 -1 1 0; 1 0 2 1]
A =
     3     2     1     0
    -1     1     2    -1
     1    -1     1     0
     1     0     2     1

>> eig(A)  % Спектр A содержит два комплексно-сопряженных числа
ans =
     2.8946
     2.0000
     0.5527+1.8694i
     0.5527-1.8694i

>> [u, t] = schur(A)  % u - матрица преобразований; t - форма Шура
```

```
u =
    0.5338   -0.8183   -0.1904   -0.0963
   -0.2296   -0.1040    0.2728   -0.9285
    0.4029    0.0344    0.9003    0.1611
    0.7071    0.5643   -0.2807   -0.3205
t =
    2.8946   -1.8580    0.9988   -0.6828
    0.0000    2.0000   -0.2150    1.6245
         0         0    0.5480    1.2470
         0         0   -2.8025    0.5575

% Преобразование к комплексной форме Шура – T
>>[U, T] = rsf2csf(u, t)
U =
    0.5338   -0.8183    0.0804-0.1057i   -0.1583+0.0535i
   -0.2296   -0.1040    0.7720+0.1514i    0.2283+0.5152i
    0.4029    0.0344   -0.1353+0.4996i    0.7487-0.0894i
    0.7071    0.5643    0.2671-0.1558i   -0.2330+0.1779i
T =
    2.8946   -1.8580    0.5666+0.5543i    0.8319+0.3789i
         0    2.0000   -1.3511-0.1193i   -0.1811-0.9015i
         0         0    0.5527+1.8694i    1.5555+0.0063i
         0         0                   0    0.5527-1.8694i

>> E = U'*U  % Проверка унитарности преобразующей матрицы U
E =
   1.0000          0.0000          0.0000+0.0000i   0.0000-0.0000i
   0.0000          1.0000          0.0000+0.0000i   0.0000+0.0000i
   0.0000-0.0000i   0.0000-0.0000i   1.0000          0.0000+0.0000i
   0.0000+0.0000i   0.0000-0.0000i   0.0000-0.0000i   1.0000

>>[P, H] = hess(A) % Построение формы Хессенберга – H
P =
    1.0000          0          0          0
         0    -0.5774    -0.7715    -0.2673
         0     0.5774    -0.1543    -0.8018
         0     0.5774    -0.6172     0.5345

H =
    3.0000    -0.5774    -1.6973    -1.3363
    1.7321     1.6667     0.0891     0.4629
         0    -1.2472     0.8333     2.5981
         0          0    -0.8660     0.5000

>> P'*P  % Проверка унитарности матрицы P: P'*P = eye(size(P))
ans =
     1.0000         0         0         0
          0    1.0000    0.0000    0.0000
          0    0.0000    1.0000    0.0000
          0    0.0000    0.0000    1.0000
```

Замечание: Более подробная информация о работе матричных функций **rref,**

rrefmovie, chol, cholinc, lu, luinc, qr, schur, rsf2csf, hess, qz и других функциях, связанных с приведением исходных матриц к различным специальным формам, дана в литературе [1–3, 5], а также содержится в обширной справочной библиотеке системы MATLAB.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ануфриев, И.Е. MATLAB 7 / И.Е. Ануфриев, А.Б. Смирнов, Е.Н. Смирнова. – СПб.: БХВ-Петербург, 2005. – 1104 с.

2. Дьяконов, В.П. MATLAB 5.0/5.3. Система символьной математики / В.П. Дьяконов, И.В. Абраменкова. – М.: Нолидж, 1999. – 633 с.

3. Кетков, Ю.Л. MATLAB 7: программирование, численные методы / Ю.Л. Кетков, А.Ю. Кетков, М.М. Шульц. – СПб.: БХВ-Петербург, 2005. – 752 с.

4. Мэтьюз, Дж. Численные методы. Использование MATLAB / Джон Мэтьюз, Куртис Финк; пер. с англ. – 3-е изд. – СПб.: Изд. дом «Вильямс», 2001. – 720 с.

5. Потемкин, В.Г. Система MATLAB. Справочное пособие / В.Г. Потемкин. – М.: ДИАЛОГ-МИФИ, 1997. – 350 с.

6. Хорн, Р. Матричный анализ / Роджер Хорн, Чарльз Джонсон; пер. с англ. – М.: Мир, 1989. – 655 с.

7. Moler, C. Numerical computing with MATLAB / Cleve Moler – http://www.mathworks.com/moler/index_ncm.html.