СИСТЕМА ИНЖЕНЕРНЫХ И НАУЧНЫХ РАСЧЕТОВ MATLAB 5.X:
В 2-Х Т. ТОМ 1.

Потемкин В. Г.

В книге дано наиболее полное описание системы MATLAB, предназначенной для выполнения инженерных и научных расчетов и высококачественной визуализации получаемых результатов. Эта система применяется в математике, вычислительном эксперименте, имитационном моделировании.

В пособии представлен исчерпывающий синтаксис команд, функций и операторов системы, элементы программирования и отладки. Система расширена новыми типами объектов: многомерными массивами, массивами записей и массивами ячеек. Существенно расширен раздел анализа и обработки данных, включая аппроксимацию, интерполяцию и геометрический анализ данных, а также численное интегрирование, решение систем обыкновенных дифференциальных уравнений, вычисление минимумов и нулей функций, преобразование Фурье, свертку и фильтрацию. Раздел линейной алгебры дополнен пакетом тестовых матриц. Существенно переработан раздел, связанный с описанием графических команд и функций, за счет использования понятий дескрипторной графики. Для удобства работы пособие снабжено индексным и предметным указателями.

Справочное пособие предназначено для инженеров, аспирантов и исследователей, выполняющих научные исследования и инженерные разработки, а также для студентов при выполнении исследовательских работ, курсовых и дипломных проектов.

ОГЛАВЛЕНИЕ

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ПРЕДИСЛОВИЕ

Эта книга предназначена для тех, кто стремится решать проблемы и задачи, возникающие в физике, химии, биологии, технике и других прикладных сферах, наиболее эффективным образом. Система инженерных и научных расчетов MATLAB широко распространена в университетах всего мира. Автору известно, что она применяется во многих технических университетах и нашей страны.

Язык, используемый в системе MATLAB, можно сравнить с языком BASIC по простоте его применения и принципу непосредственного исполнения (интерпретации). Ориентация на работу с массивами делает его удобным и естественным инструментом обработки экспериментальных данных. Так же, как и язык BASIC, который постепенно трансформировался в объектно-ориентированный язык Visual Basic, язык MATLAB стремится стать универсальным языком технических вычислений. Об этом свидетельствует создание компилятора языка и математических библиотек на языках C и C++, включение таких элементов объектно-ориентированного программирования, как дескрипторы графических объектов, понятие классов и методов их переопределения.

На решениях математических задач с помощью системы MATLAB следует остановиться особо. Будучи ориентированной на работу с реальными данными, эта система выполняет все вычисления в арифметике с плавающей точкой в отличие от систем компьютерной алгебры REDUCE, MACSYMA, DERIVE, Maple, Mathematica, Theorist, где преобладает целочисленное представление и символьная обработка данных. Каждое из этих направлений характеризуется присущими только ему методами и алгоритмами решения задач, а их перенос из одной среды в другую может быть затруднительным, а зачастую и невозможным. Поэтому для решения проблем на стыке символьных вычислений и вычислений с плавающей точкой в состав интегрированной системы MATLAB включен пакет прикладных программ Extended Symbolic Mathematics Toolbox, реализующий интерфейс с системой символьных вычислений Maple. Намечается тенденция к включению элементов распараллеливания на уровне М-файлов. С этой целью фирмой Alpha Data Parallel Systems, Ltd. (Великобритания) предложены аппаратные и программные решения с использованием ускорительных плат на базе процессоров Alpha AXP и специального программного пакета ParaMat. Это означает, что применение алгоритмов с естественным распараллеливанием может быть реализовано в рамках операционной среды системы MATLAB.

В данном справочном пособии представлены операторы, функции и команды языка MATLAB, классифицированные по областям применения -линейная алгебра, анализ и обработка данных, вычисления с разреженными матрицами, визуализация результатов и презентационная графика. Для удобства книга снабжена индексным указателем, который включает более 800 конструкций языка.

Наблюдаемая тенденция развития языка MATLAB связана с включением в его состав таких объектов, как многомерные массивы, массивы элементов разных типов, структуры, и превращением его в объектно-ориентированный язык с возможностью переопределения классов и методов. Это, с одной стороны, усложняет язык программирования, позволяя разработчикам прикладных систем создавать эффективные и независимо исполняемые приложения, с другой стороны, упрощает работу с отдельными разделами за счет создания разработчиками системы MATLAB специализированных интерфейсов, которые предоставляют пользователю удобные средства выбора и правильного применения специфических команд и функций этого раздела. Примерами таких разработок являются решатели дифференциальных уравнений, диалоговый графический редактор, средства написания электронных учебников.

ВВЕДЕНИЕ

Система MATLAB (сокращение от MATrix LABarotatory - МАТричная ЛАБоратория) разработана фирмой The MathWorks, Inc. (США, г. Нейтик, шт. Массачусетс) и является интерактивной системой для выполнения инженерных и научных расчетов, которая ориентирована на работу с массивами данных. Система использует математический сопроцессор и допускает возможность обращения к программам, написанным на языках Fortran, C и C++.

Система поддерживает выполнение операций с векторами, матрицами и массивами данных, реализует сингулярное и спектральное разложения, вычисление ранга и чисел обусловленности матриц, поддерживает работу с алгебраическими полиномами, решение нелинейных уравнений и задач оптимизации, интегрирование в квадратурах, решение дифференциальных и разностных уравнений, построение различных видов графиков, трехмерных поверхностей и линий уровня. В системе реализована удобная операционная среда, которая позволяет формулировать проблемы и получать решения в привычной математической форме, не прибегая к рутинному программированию.

Наиболее известные области применения системы MATLAB:

- математика и вычисления;
- разработка алгоритмов;
- вычислительный эксперимент, имитационное моделирование;
- анализ данных, исследование и визуализация результатов;
- научная и инженерная графика;
- разработка приложений, включая графический интерфейс пользователя.

MATLAB - это интерактивная система, основным объектом которой является массив, для которого не требуется указывать размерность явно. Это позволяет решать многие вычислительные задачи, связанные с векторно-матричными формулировками, существенно сокращая время, необходимое для программирования на скалярных языках типа C или Fortran.

Система MATLAB распространяется в течение почти 15 лет, она постоянно совершенствуется благодаря активному сотрудничеству пользователей с фирмой The MathWorks, Inc. - разработчиком системы. В университетском мире эта система широко используется в преподавании не только курсов по вычислительным методам линейной алгебры, но и специальных курсов, ориентированных на применение методов оптимизации, аппроксимации, идентификации, анализа систем. В промышленных приложениях система применяется для численных расчетов, макетирования алгоритмов, исследований в области автоматического управления, статистической обработки сигналов и процессов.

Система MATLAB - это одновременно и операционная среда и язык программирования. Одна из наиболее сильных сторон системы состоит в том, что на языке MATLAB могут быть написаны программы для многократного использования. Пользователь может сам написать специализированные функции и программы, которые оформляются в виде М-файлов. По мере увеличения количества созданных программ возникают проблемы их классификации и тогда можно попытаться собрать родственные функции в специальные папки. Это приводит к концепции пакетов прикладных программ (ППП), которые представляют собой коллекции М-файлов для решения определенной задачи или проблемы.

В действительности ППП - это нечто большее, чем просто набор полезных функций; часто это результат работы многих исследователей по всему миру, которые объединяются в группы по интересам. Именно поэтому пакеты прикладных программ MATLAB Application Toolboxes, входящие в состав семейства продуктов MATLAB, позволяют находиться на уровне самых современных мировых достижений.

Системы семейства MATLAB 5 включают следующие программные продукты, сопровождаемые справочной и методической документацией:

Базовые программные средства

№ п/п	*Наименование продукта*	*Версия*		*Назначение*
		5.2	*5.0*	
1	MATLAB for Windows	5.2	5.0	Система инженерных и научных расчетов
2	MATLAB Compiler	1.2	1.1	Компилятор языка MATLAB на язык С
3	MATLAB C Math Library	1.2	1.1	Библиотека математических функций системы MATLAB на языке С
4	MATLAB C++ Math Library	1.2	1.0	Библиотека математических функций системы MATLAB на языке С++
5	SIMULINK for Windows	2.2	2.0	Моделирование динамических систем
6	Real-Time Workshop (RTW)	2.2	1.1c	Моделирование систем реального времени
7	SIMULINK RTW Ada Extension	1.2	1.1c	Расширение RTW на базе языка Ada
8	SIMULINK Accelerator	2.2	1.1a	Ускоритель процедур моделирования
9	Applix Link	1.0.2	-	Интерфейс с семейством продуктов Applixware фирмы Applix для платформ UNIX и WindowsNT

10	Excel Link	1.0	1.0	Интерфейс с системой Excel 5.0 и выше
11	The Student Edition of MATLAB	5	5	Версия MATLAB для студентов
12	The Student Edition of SIMULINK	2.0	2.0	Версия SIMULINK для студентов

Пакеты прикладных программ

№ п/п	*Наименование продукта*	*Версия*		*Назначение*
		5.2	*5.0*	
	Математика			
1	NAG Foundation Toolbox	1.0.2	1.0	Библиотека математических функций The Numerical Algorithms Group Ltd.
2	Spline Toolbox	2.0	1.1.3	Сплайн-аппроксимация
3	Statistics Toolbox	2.1.1	2.0.1	Статистика
4	Optimization Toolbox	1.5.2	1.5.0	Оптимизация
5	Fuzzy Logic Toolbox	2.0	1.0.2	Размытые множества
6	Neural Network Toolbox	3.0	2.0.3	Нейронные сети
7	Partial Differential Equations Toolbox	1.0.3	1.0.1	Уравнения в частных производных
8	Symbolic Math Toolbox	2.0.1	2.0	Символьная математика
9	Extended Symbolic Math Toolbox	2.0.1	2.0	Расширенная символьная математика (включает систему Maple)
	Анализ и синтез систем управления			
10	Control System Toolbox	4.1	4.0	Системы управления
11	Nonlinear Control Design Toolbox	1.1.2	1.1.0	Проектирование нелинейных систем
12	Robust Control Toolbox	2.0.5	2.0.3	Робастное управление
13	Model Predictive Control Toolbox	1.0.3	1.0.1	Управление с эталонной моделью
14	μ-Analysis and Synthesis Toolbox	3.0.3	3.0.1	μ-анализ и синтез
15	Quantitative Feedback Theory Toolbox	1.0.3	1.0.1	Проектирование робастных систем с обратной связью
16	LMI (Linear Matrix Inequality) Control Toolbox	1.0.4	1.0.1	Синтез систем управления на основе линейных матричных неравенств

	Идентификация систем управления			
17	System Identification Toolbox	4.0.4	4.0.1	Идентификация параметров
18	Frequency Domain System Identification Toolbox	2.0.3	2.0.1	Идентификация в частотной области
	Моделирование с использованием SIMULINK			
19	Data Signal Processing (DSP) Blockset	2.2	1.0	Обработка цифровых сигналов
20	Fixed Point Blockset	1.2	1.0	Фиксированная разрядность
21	Power System Blockset	1.0	-	Моделирование энергетических систем
22	Stateflow Toolbox	1.0.6	-	Моделирование событий
	Обработка сигналов и изображений			
23	Signal Processing Toolbox	4.1	4.0	Обработка сигналов
24	Higher-Order Spectral Analysis Toolbox	2.0.2	2.0.1	Спектральный анализ с учетом моментов высшего порядка
25	Image Processing Toolbox	2.1	2.0	Обработка изображений
26	Wavelet Toolbox	1.1	1.0.2	Импульсная декомпозиция
	Разное			
27	Communication Toolbox	1.3	1.0.1	Системы связи и коммуникаций
28	Financial Toolbox	1.1	1.0.2	Финансы
29	Mapping Toolbox	1.0.1	-	Картография
30	Database Toolbox	1.0	-	Работа с базами данных
31	Notebook Toolbox			Написание М-книг

ППП The MATLAB Notebook - это специальное средство динамического интерфейса системы MATLAB с текстовым редактором Microsoft Word. Оно позволяет создавать интерактивные текстовые документы, которые содержат исполняемые команды системы MATLAB и графический вывод.

Используя ППП Notebook - записную книжку, можно создавать М-книги, которые, с одной стороны, являются обычными документами текстового редактора Microsoft Word, а с другой - содержат не только текст, но и команды системы MATLAB, а также результаты их выполнения. Можно рассматривать М-книгу как сценарий на языке MATLAB, выполняемый в среде редактора Word с полным набором мощных средств редактирования. Если же вы являетесь пользователем редактора Microsoft Word, то вы обнаружите, что Notebook - это естественная среда, в которой объединяются удобные и мощные возможности редактора с вычислительными и графическими

возможностями системы MATLAB. ППП Notebook позволяет создавать интерактивную техническую документацию в виде:

- электронных книг;
- электронных справочников;
- интерактивных сценариев;
- технических отчетов и альбомов проектирования;
- электронных записных книжек;
- электронных отчетов по домашним заданиям, курсовому и дипломному проектированию.

При написании справочного пособия использована следующая документация по системам семейства MATLAB 5:

- MATLAB 5.2 Product Family New Features;
- MATLAB Function Reference;
- Using MATLAB;
- Using MATLAB Graphics;
- MATLAB Notebook User's Guide.

1. ОПЕРАЦИОННАЯ СРЕДА СИСТЕМЫ MATLAB 5

Операционная среда системы MATLAB 5 - это множество интерфейсов, которые поддерживают связь этой системы с внешним миром. Сюда входят диалог с пользователем через командную строку или графический интерфейс, просмотр рабочей области и путей доступа, редактор и отладчик М-файлов, работа с файлами и оболочкой DOS, экспорт и импорт данных, интерактивный доступ к справочной информации, динамическое взаимодействие с внешними системами, такими, как Microsoft Word, Excel и др.

Реализуются эти интерфейсы через командное окно, инструментальную панель, подсистемы просмотра рабочей области и путей доступа, редактор/отладчик М-файлов, специальные меню.

Командное окно

Командное окно системы MATLAB показано на рис. 1.1. Здесь же показано ниспадающее меню File.

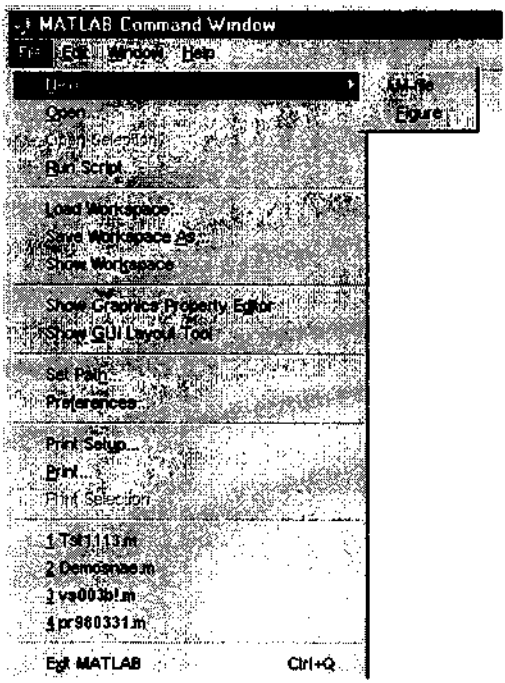


Рис. 1.1

Оно содержит следующие опции:

Опция	*Подопции*	*Назначение*
New	M-file Figure	Открыть в редакторе/отладчике новый файл Открыть графическое окно
Open		Открыть в редакторе/отладчике указанный файл
Open Selection		Открыть в редакторе/отладчике файл, выделенный в строке командного окна
Run Script		Вызов окна для запуска Script-файла
Load Workspace		Вызов окна загрузки MAT-файла
Save Workspace As		Вызов окна сохранения MAT-файла
Show Workspace		Вызов средства просмотра рабочей области Workspace Browser
Show Graphics Property Editor *		Вызов редактора свойств графических объектов
Show GUI Layout Tool *		Вызов средств разработки графического интерфейса пользователя
Set Path		Вызов средства просмотра путей доступа Path Browser
Preferences		Выбор характеристик
Print Setup		Установка опций принтера
Print		Установка опций вывода на печать
Print Selection		Печать выделенного фрагмента

Примечание. Опции, помеченные знаком *, доступны начиная с версии MATLAB 5.2.

Особого рассмотрения заслуживает опция Preferences (Выбор характеристик), которая включает 3 окна. В первую очередь рассмотрим окно General (Общее) (рис. 1.2).

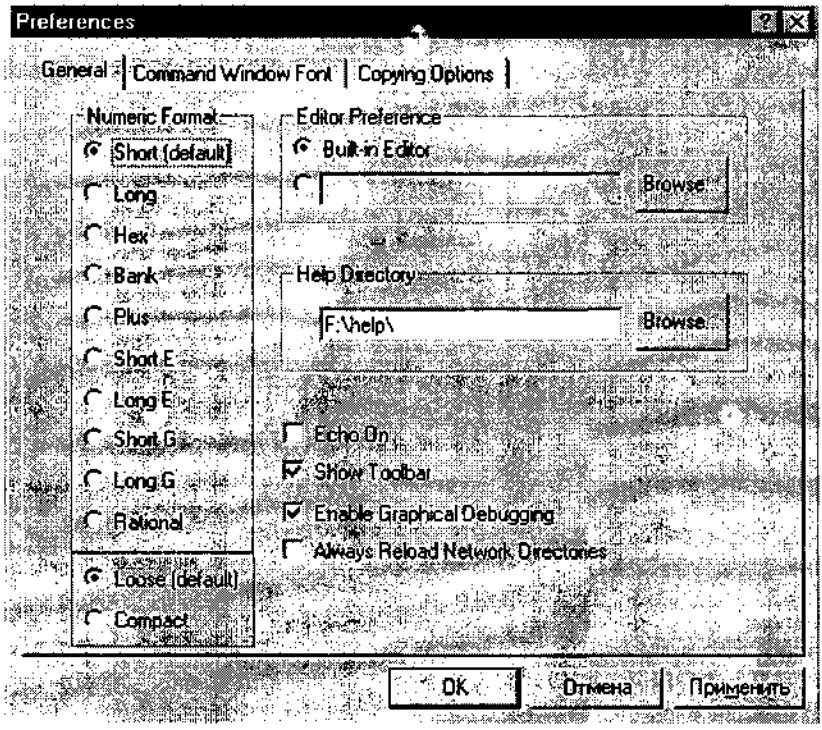


Рис. 1.2

В этом окне можно видеть 3 поля и 3 маркера, имеющие следующие назначения:

Формат данных	*Назначение*
Numeric Format	Выбор формата представления чисел и межстрочного пробела. По умолчанию формат Short, пробел Loose
Editor Preference	Выбор текстового редактора По умолчанию встроенный редактор Built in Editor
Help Directory	Каталог справки Help
Echo on	Показывать на экране команды исполняемого Script-файла сценария/Не показывать
Show Toolbar	Показывать на экране инструментальную панель/Не показывать
Enable Graphical Debugging	Поддерживать отладку графики/Не поддерживать

Далее рассмотрим окно Command Window Font (Шрифт для командного окна) (рис. 1.3).

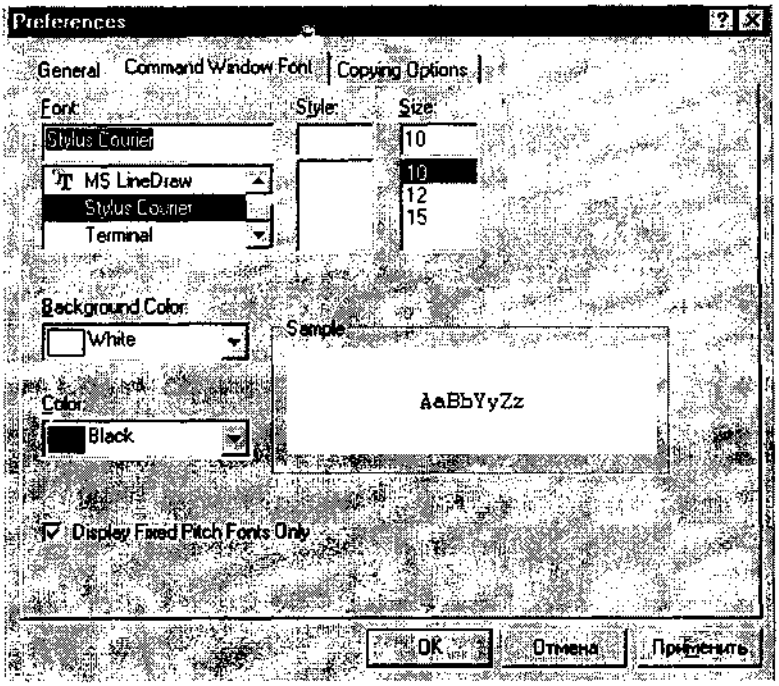


Рис. 1.3

В этом окне можно видеть 6 полей и 1 маркер, имеющие следующие назначения:

Поле или маркер	*Назначение*	
Font	Шрифт для вывода текста в командном окне	
Style	Тип шрифта:	Light - светлый Regular - нормальный Bold - жирный
Size	Размер шрифта:	10 12 15
Background Color	Цвет фона:	Silver - серебристый Red - красный Lime - лимонный Yellow - желтый Blue - синий Fuscia - светло-фиолетовый Aqua - голубой White - белый

Color	Цвет символа:	Black - черный Maroon - каштановый Green - зеленый Olive - оливковый Navy - темно-синий Purple - темно-фиолетовый Teal - зелено-голубой Gray - серый
Sample	Образец фона и шрифта	
Display Fixed Pitch Fonts Only	Показать только шрифты с фиксированным шагом/Показать все шрифты	

Наконец, рассмотрим окно Copying Options (Опции копирования) (рис. 1.4).

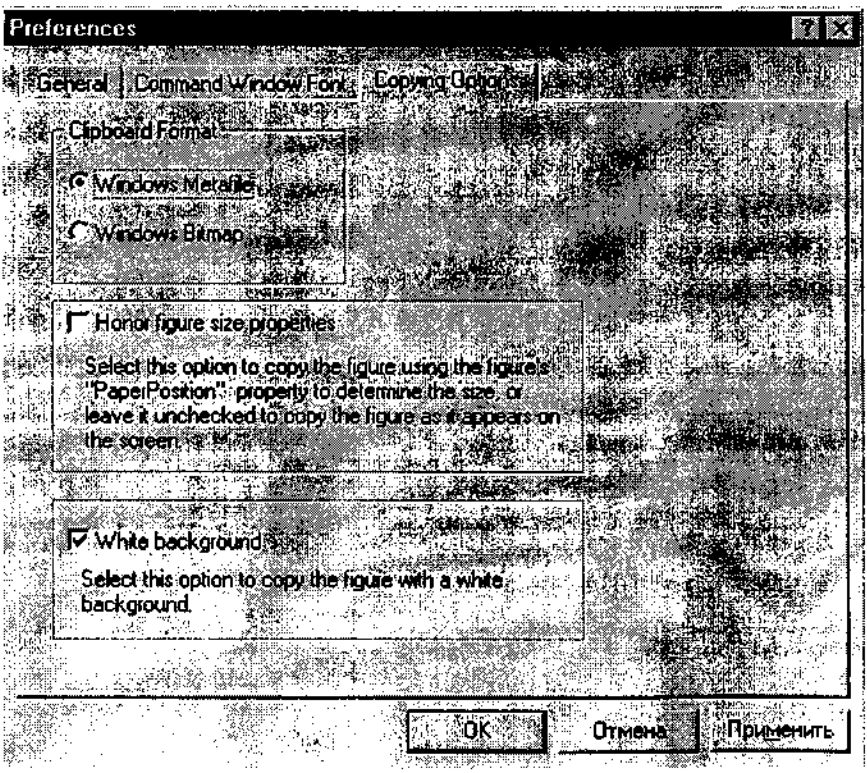


Рис. 1.4

В этом окне можно видеть 3 поля, имеющие следующие назначения:

Поле или маркер	*Назначение*
Clipboard Format	Формат копирования в буфер обмена: Windows Metafile Windows Bitmap
Honor figure size properties	Маркер воспроизведения размеров рисунка. Выбор этой опции позволяет копировать рисунок с учетом свойства "Paper Position" (для формата Windows Bitmap не действует)
White Background	Маркер белого/черного фона (для формата Windows Bitmap не действует)

Инструментальная панель

Инструментальная панель командного окна системы MATLAB позволяет обеспечить простой доступ к операциям над М-файлами (рис. 1.5).

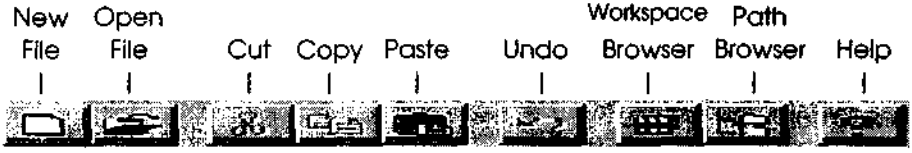


Рис. 1.5

Эти операции включают:

- создание нового М-файла (New File);
- открытие существующего М-файла (Open File);
- удаление фрагмента (Cut);
- копирование фрагмента (Copy);
- вставку фрагмента (Paste);
- восстановление только выполненной операции (Undo);
- просмотр рабочей области (Workspace Browser);
- просмотр путей доступа (Path Browser);
- текущую помощь (Help).

Редактор/отладчик М-файлов

В состав системы MATLAB 5 входит редактор/отладчик М-файлов M-file Editor/Debugger, который может быть вызван из командной строки командой edit или edit <*имя М-файла*>. Инструментальная панель командного окна этого редактора/отладчика для версии 5.2 показана на рис. 1.6.

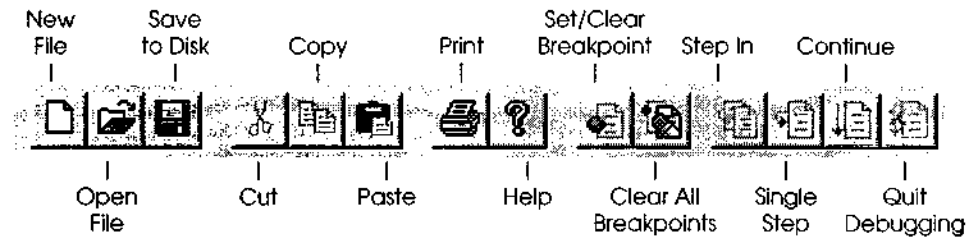


Рис. 1.6

Редактор/отладчик поддерживает следующие операции:

- создание нового М-файла (New File);
- открытие существующего М-файла (Open File);
- сохранение М-файла на диске (Save to Disk);
- удаление фрагмента (Cut);
- копирование фрагмента (Copy);
- вставка фрагмента (Paste);
- печать М-файла (Print);
- текущая помощь (Help);
- установить/удалить контрольную точку (Set/Clear Breakpoint);
- удалить все контрольные точки (Clear All Breakpoints);
- войти в М-модуль (Step In);
- выполнить один шаг отладки (Single Step);
- продолжить выполнение (Continue);
- завершить отладку (Quit Debugging).

Рабочая область

Рабочая область системы MATLAB - это область памяти, в которой размещены переменные системы. Содержимое этой области можно просмотреть из командной строки с помощью команд who и whos. Команда who выводит только имена переменных, а команда whos - информацию о размерах массивов и типе переменной.

Рассмотрим в качестве примера 5 массивов различного типа:

- A - трехмерный массив чисел удвоенной точности;
- B - массив разреженной структуры;
- C - массив ячеек;
- D - массив записей;
- S - массив символов.

```
whos
  Name          Size          Bytes      Class

  A             4x3x2           192      double array
  B             4x5              72      sparse array
  C             4x3x2          2400      cell array
  D             4x2            2496      struct array
  S             4x5              40      char array

Grand total is 144 elements using 5200 bytes
```

Специальное средство просмотра Workspace Browser обеспечивает представление команды whos в виде графического интерфейса. Для того чтобы открыть Workspace Browser, надо либо выбрать опцию Show Workspace из меню File menu, либо воспользоваться кнопкой Workspace Browser инструментальной панели.

В результате этих операций на терминал будет выведено окно (рис. 1.7).

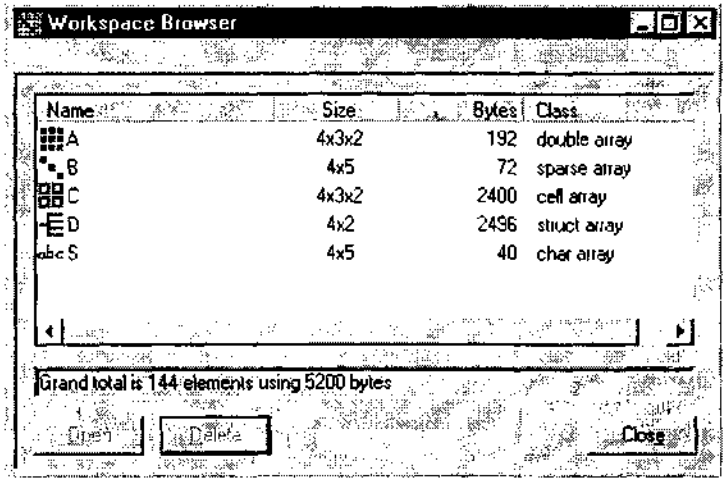


Рис. 1.7

В этом окне можно выполнить следующие операции:

- удалить переменную, если выделить ее и нажать кнопку Delete;
- закрыть окно с помощью кнопки Close.

Кроме того, можно изменять размеры колонок посредством перемещения их границ с помощью мыши. Можно выполнить переименование переменной, если сначала выделить ее, затем однократно щелкнуть левой клавишей мыши (заметим, что двойной щелчок никакого действия не оказывает). После короткой задержки появляется поле, в котором можно указать новое имя; и наконец, следует нажать клавишу Enter, чтобы подтвердить завершение операции.

Загрузка и сохранение рабочей области

Команды save и load позволяют в любой момент времени сохранить содержимое рабочей области или загрузить новые данные в процессе сеанса работы. С помощью этих команд можно также осуществлять экспорт и импорт ASCII-файлов.

Сохранение переменных рабочей области. Команда save позволяет сохранить содержимое рабочей области в двоичном MAT-файле, который можно в дальнейшем вызвать командой load. Команда save также доступна в качестве опции Save Workspace меню File.

Спецификация формата файла. Для того чтобы управлять форматами файлов, следует в команде save в дополнение к имени файла и списку переменных использовать следующие флаги:

Флаг	*Пояснение*
–mat	Двоичный MAT-файл (по умолчанию)
–ascii	ASCII-формат (8 цифр)
–ascii –double	ASCII-формат (16 цифр)
–ascii –double –tabs	Формат с разделителями и метками табуляции
–v4	Формат версии MATLAB 4
–append	Добавить данные к существующему MAT-файлу

При использовании флага v4 можно сохранить только те данные, которые совместимы с данными, используемыми в версии MATLAB 4; это означает, что сохранить такие типы данных, как массивы записей, ячеек, многомерные массивы или объекты, нельзя.

Когда содержимое рабочей области сохраняется в ASCII-формате, то рекомендуется единовременно сохранять только одну переменную. Если сохраняется более одной переменной, то система MATLAB создаст ASCII-файл, который нельзя будет в дальнейшем загрузить в MATLAB, используя команду load.

Загрузка рабочей области. Команда load позволяет загрузить MAT-файл, который был ранее сохранен с помощью команды save. При загрузке MAT-файла новые значения одноименных переменных будут записаны взамен старых.

Если MAT-файл имеет расширение, отличающееся от .mat, то необходимо использовать флаг –mat; в противном случае MATLAB будет считать форматом файла ASCII-формат.

Загрузка файлов данных в ASCII-формате. Команда load позволяет выполнять импорт файлов данных в ASCII-формате; она преобразовывает содержимое файла в переменную с именем файла только без расширения.

Например, применение команды load tides.dat создает в рабочей области системы MATLAB переменную с именем tides. Если исходный файл в ASCII-формате имеет m lines строк с n значениями в каждой строке, то результатом будет массив чисел размера m×n.

Использование имен в формате строк. Если имена файлов и переменных представляют собой строковые переменные, то можно, используя свойство дуальности команды и функции, рассматривать команды load и save как функции. В этом случае входные переменные должны следовать в том же порядке, в каком они следовали в командной строке.

Например, последовательность операторов

```
save('myfile', 'VAR1', 'VAR2')
A = 'myfile';
load(A)
```

это то же самое, что и последовательность команд

```
save myfile VAR1 VAR2
load myfile
```

Для сохранения или загрузки последовательности файлов, имена которых имеют общий корень и дополнительный целочисленный суффикс, необходимо использовать структуру цикла.

Например, следующая конструкция позволяет сохранить квадраты чисел от 1 до 10 в файлах с именами data1, ..., data10:

```
file = 'data';
for i = 1:10
j = i.^2;
save([file int2str(i)],'j');
end
```

Использование группового символа. Команды load и save допускают использование группового символа * в качестве замены ряда символов в шаблоне имени переменной.

Например, команда save rundate x* сохраняет все переменные, имена которых начинаются с символа x в файле с именем rundata.mat.

Точно так же команда load testdata ex1*95 загружает все переменные, имена которых начинаются с символов 'ex1' и заканчиваются символами '95', независимо от того, какие символы размещены между ними.

Список путей доступа

Для поиска М-файлов система MATLAB использует механизм путей доступа, поскольку М-файлы записываются в каталоги или папки файловой системы.

Например, при поиске файла с именем foo MATLAB выполняет следующие действия:

- просматривает, не является ли foo именем переменной;
- просматривает, не является ли foo встроенной функцией;
- ищет в текущем каталоге М-файл с именем foo.m;
- ищет М-файл с именем foo.m во всех каталогах списка путей доступа.

Реально применяемые правила поиска являются более сложными из-за ограничений, которые связаны с использованием подфункций, частных функций и объектно-ориентированных механизмов. Однако приведенный выше упрощенный порядок поиска точно отражает механизм поиска М-файлов, с которыми обычно работает пользователь.

Работа со списком путей доступа. В процессе сеанса работы можно вывести на терминал или внести изменения в список путей доступа, используя следующие функции:

- path выводит на экран списка путей доступа;
- path(s) заменяет существующий список списком s;
- addpath /home/lib и path(path, '/home/lib') добавляют новый каталог в список путей доступа;
- rmpath /home/lib удаляет путь /home/lib из списка.

Список путей доступа, используемый по умолчанию, определен в файле pathdef.m, который размещен в каталоге local; этот файл выполняется при каждом запуске системы MATLAB.

Кроме работы из командной строки существует средство просмотра путей доступа Path Browser, которое поддерживает удобный графический интерфейс для просмотра и изменения списка путей. Однако более предпочтительно вносить непосредственные изменения в М-файл pathdef.m, используя какой-либо текстовый редактор, в том числе и редактор/отладчик системы MATLAB.

Текущий каталог. Система MATLAB использует понятие *текущего каталога* при работе с М- и МАТ-файлами во время сеанса работы. *Начальный текущий каталог* определен в файле запуска, который ассоциирован с ярлыком запуска системы MATLAB, расположенном на рабочем столе. Щелчок правой кнопки мыши, установленной на этом ярлыке, и выбор опции Properties позволяют изменить начальный каталог, используемый по умолчанию.

Для вывода текущего каталога на экран терминала предназначена команда cd. Для изменения текущего каталога следует использовать команду cd <*новый путь доступа*>.

Просмотр списка файлов. Мы уже видели, как команда path позволяет отобразить список путей доступа. В свою очередь, команда what позволяет увидеть список файлов, расположенных в заданном или текущем каталоге.

Команда what без параметров выводит на экран список файлов текущего каталога, а команда what <*полный или частичный путь доступа*> выводит на экран список файлов заданного каталога.

Для распечатки содержимого М-файла предназначена команда type <*имя файла*>; для редактирования М-файла используется команда edit <*имя файла*>.

Средство просмотра путей доступа. На платформе PC имеется средство визуального просмотра путей доступа Path Browser, которое позволяет просматривать, модифицировать пути доступа и видеть списки всех файлов системы MATLAB. Для того чтобы открыть средство просмотра Path Browser, следует использовать либо опцию Set Path из меню File, либо кнопку инструментальной панели Path Browser.

Окно средства просмотра путей доступа Path Browser показано на рис. 1.8.

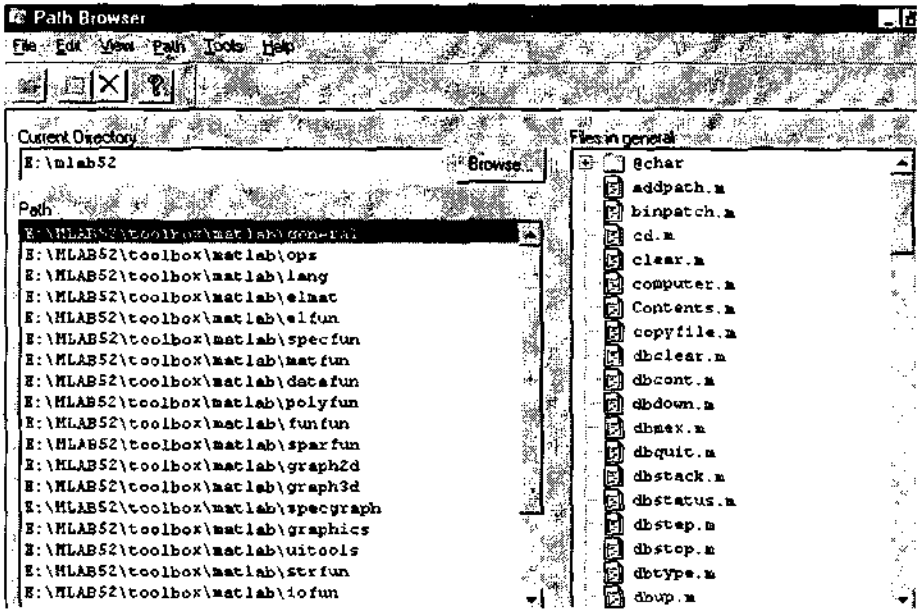


Рис. 1.8

В этом окне имеются:

- поле Current Directory с кнопкой Browse, предназначенное для изменения текущего каталога;
- поле Path - содержит список путей доступа;
- поле Files in <*имя каталога, выделенного в поле Path*> - содержит список файлов и внутренних каталогов типа private, @;
- инструментальная панель:

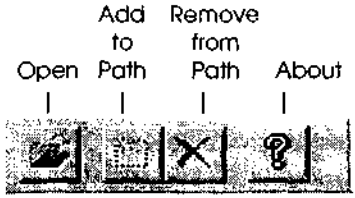


Рис. 1.9

- меню приложения Path Browser:

Имя меню	*Имя подменю*	*Пояснение*
File	Open	Открыть файл
	Save Path	Сохранить путь доступа
	Exit Path Browser	Выйти из приложения
Edit	Undo	Отменить последнюю операцию
	Redo	Вернуть отмененную операцию
View	Toolbar	Показать инструментальную панель
	Status Bar	Показать панель состояния
	Editor Debugger	Показать редактор/отладчик
	Path Browser	Показать Path Browser
	Workspace Browser	Показать Workspace Browser
Path	Add to Path	Добавить каталог к пути доступа
	Remove from Path	Удалить каталог из пути доступа
	Refresh	Обновить путь доступа
	Restore Defaults	Восстановить установки, принятые по умолчанию
Tools	Run	Выполнить М-файл
	Customize	Выбрать настройки
	Options	Установить опции
	Font	Установить шрифт
Help	About Matlab Path Browser	Информация о Path Browser

Для перемещения каталога в другую позицию в области Path следует захватить его левой кнопкой мыши и переместить на нужное место.

Если изменение списка путей доступа выполняется в командном окне, то для отражения этих изменений в средстве просмотра Path Browser необходимо использовать команду Refresh из меню Path.

Все изменения, которые вносятся в список путей доступа, действуют только в течение сеанса работы; для того чтобы внести их в файл pathdef.m для постоянного использования, необходимо воспользоваться командой Save Path из меню File.

Работа с файлами и оболочкой DOS

Команды cd, dir, delete, type позволяют из командной строки системы MATLAB выполнить ряд команд DOS, связанных с управлением файлами. Приведенная таблица отражает связь команд системы MATLAB с командами DOS:

MATLAB	MS-DOS
cd	chdir
dir	dir
delete	del или erase
type	type

Большинство этих команд позволяет указывать пути доступа, имена дисководов, использовать групповые символы.

Запуск внешних программ. Признаком перехода к выполнению команд DOS является знак !, который указывает, что следующая за ним команда - это команда DOS. Это исключительно полезно при вызове утилит и выполнении внешних других программ без выхода из системы MATLAB.

Импорт и экспорт данных

Существует много приемов для перемещения данных между системой MATLAB и другими приложениями. В большинстве случаев при работе с данными системы MATLAB можно просто использовать команды чтения и записи файлов. Для более сложных наборов данных можно создать собственные программы для чтения и записи на языках C или Fortran.

Импортирование данных в систему MATLAB. Существует несколько способов для передачи данных из других приложений в систему MATLAB. Выбор способа зависит от объема и формата данных.

- **Ввод данных в виде списка.** Если количество данных невелико, то их можно просто напечатать, помещая в квадратные скобки. Этот метод неудобен при большом количестве данных, поскольку их невозможно редактировать.
- **Формирование данных в М-файле.** Используя текстовый редактор, можно сформировать М-файл, в котором данные представлены как список элементов. По существу, это тот же первый способ, но он имеет то преимущество, что позволяет с помощью редактора корректировать данные. Достаточно после исправления перезапустить М-файл, чтобы ввести исправленные данные.
- **Загрузка данных из ASCII-файла.** ASCII-файлы накапливают данные в семиразрядном коде без контроля по четности. Каждая строка содержит одинаковое количество значений, разделенных пробелами, и завершается

символом возврата каретки. Эти файлы можно редактировать, используя обычный текстовый редактор. Их можно читать непосредственно в системе MATLAB, используя функцию load. При этом создается переменная, имя которой совпадает с именем файла. Можно воспользоваться функций dlmread, чтобы указать другой тип разделителя.

- **Чтение данных с использованием функций ввода/вывода.** Применение функций ввода/вывода, а также функций fopen и fread полезно при загрузке файлов данных из других приложений, использующих специальные форматы данных.
- **Использование специальных средств для чтения файлов.** Для чтения файлов, записанных в специальных форматах, в системе MATLAB имеются следующие специализированные функции:

Функция	*Назначение*
dlmread	Чтение ASCII-файлов
wk1read	Чтение электронных таблиц в формате WK1
imread	Чтение изображения из графического файла
auread	Чтение звукового файла с расширением .au (формат фирмы SUN Microsystems)
wavread	Чтение звукового файла с расширением .wav (формат фирмы Microsoft)

- **Создание MEX-файла.** Наилучший способ создания программ для чтения данных - это использовать уже имеющиеся программы на языках C или Fortran для чтения данных из других приложений. Однако этот метод, называемый *смешанным программированием*, требует написания специальных программ-связок, оформляемых в виде MEX-файлов.
- **Разработка программы на языках Fortran или C.** Программисты, использующие языки Fortran или C, могут написать специальные программы для преобразования данных в формат MAT-файла системы MATLAB. В этом случае преобразованные данные могут быть загружены в систему MATLAB с помощью обычной команды load.

Экспортирование данных из системы MATLAB. Существует несколько способов для передачи данных из системы MATLAB в другие приложения.

- **Использование команды diary.** Для массивов небольших размеров можно использовать команду diary, чтобы создать файл дневника, который включает команды MATLAB, используемые в течение сеанса работы, а также позволяет на экране просмотреть необходимые данные. Записи дневника могут быть полезны для вложения в документы или отчеты. В дальнейшем можно использовать текстовый редактор для редактирования дневника.

- **Сохранение данных в формате ASCII.** Команда save с опцией -ascii позволяет записать данные в этом формате, причем, используя команду dlmwrite, можно задать другой тип разделителя.
- **Использование специальных средств для записи файлов.** Для записи файлов в специальных форматах, определяемых приложениями, в системе MATLAB имеются следующие специализированные функции:

Функция	*Назначение*
dlmwrite	Запись данных в ASCII-файл
wk1write	Запись данных в электронную таблицу в формате WK1
imwrite	Запись изображения в графический файл
auwrite	Запись данных в звуковой файл с расширением .au (формат фирмы SUN Microsystems)
wavwrite	Запись данных в звуковой файл с расширением .wav (формат фирмы Microsoft)

- **Создание MEX-файла.** Наилучший способ создания программ для записи данных - это использовать уже имеющиеся программы на языках C или Fortran для записи данных в другие приложения. Однако этот метод, называемый *смешанным программированием*, требует написания специальных программ-связок, оформляемых в виде MEX-файлов.
- **Разработка программы на языках Fortran или C.** Программисты, использующие языки Fortran или C, могут написать специальные программы для преобразования данных из формата MAT-файла системы MATLAB в формат приложения. В этом случае данные могут быть выгружены из системы MATLAB с помощью обычной команды save.
- **Текстовые файлы с разделителями.** Функции dlmread и dlmwrite позволяют читать и записывать данные, отделенные разделителем, используя ASCII-файл. В качестве разделителя может быть использован любой символ, который отделяет одно значение от другого.

Например, рассмотрим файл с именем ph.dat, который содержит данные, разделенные точкой с запятой:

```
7.2;8.5;6.2;6.6
5.4;9.2;8.1;7.2
```

Для того чтобы прочитать содержимое этого файла в массив с именем A, надо использовать следующий оператор:

```
A = dlmread('ph.dat', ';');
```

Второй аргумент функции dlmread указывает тип разделителя.

В дополнение к разделителю, который вы используете, функция dlmread также считает разделителями имеющиеся пробелы. Поэтому функция dlmread, приведенная выше, будет работать правильно, если даже содержимое файла ph.dat будет таким:

```
7.2;    8.5;    6.2;6.6
5.4;    9.2;    8.1;7.2
```

Предупреждение:

Первый аргумент М-функции dlmread - это имя файла, а не идентификатор файла. Поэтому не надо предварительно открывать файл с помощью функции fopen, а следует сразу применять функции dlmread и dlmwrite.

Продемонстрируем, как функция dlmwrite выполняет запись текста с разделителями во внешний файл с именем myfile, используя разделитель ";":

```
A =
1 2 3
4 5 6
dlmwrite('myfile',A,';')
1;2;3
4;5;6
```

Обмен файлами данных для различных платформ. Иногда оказывается необходимо работать с версиями системы MATLAB для разных вычислительных платформ или передавать разработанные приложения на другие системы. Приложения, создаваемые в системе MATLAB, могут включать М-файлы, представляющие собой М-функции или М-сценарии, а также MAT-файлы, содержащие двоичные данные. Оба типа файлов могут быть непосредственно использованы на различных платформах:

- М-файлы являются ASCII-файлами, содержащими обычный текст. Они независимы от типа используемого компьютера. В то же время для различных платформ символами окончания строки могут быть как символ CR, так и символ LF. Интерпретатор системы MATLAB допускает любые комбинации.
- MAT-файлы являются двоичными файлами и зависят от типа используемого компьютера. Тем не менее они могут переноситься с одного типа компьютера на другой, поскольку содержат признак используемого компьютера в заголовке файла. Система MATLAB проверяет этот признак, когда загружает файл, и, если оказывается, что файл создан на компьютере другой платформы, выполняет необходимое преобразование.

Чтобы использовать MATLAB на компьютерах различных платформ, необходимы программы обмена данными для двоичного и ASCII-формата. При использовании этих программ надо быть уверенным, что MAT-файлы передаются как двоичные файлы, М-файлы - как ASCII-файлы. Ошибка в установке соответствующих режимов обычно разрушает данные.

Команда diary. Эта команда позволяет сформировать дневник сеанса работы, включая графический вывод. Дневник записывается в специальный файл на жестком диске. После сеанса работы этот файл можно просмотреть с помощью любого текстового редактора.

Например, чтобы создать в текущем каталоге файл дневника с именем febr01.out, следует использовать команду diary febr01.out.

Для того чтобы в процессе ведения дневника прервать запись, достаточно воспользоваться командой diary off, а для возобновления - командой diary on.

М-файл Startup. Файл matlabrc.m, который размещен в каталоге local, зарезервирован для использования программистами фирмы MathWorks, а на многопользовательских системах - для использования менеджером системы.

Файл startup.m предназначен для пользователя. В нем можно установить задаваемые по умолчанию пути доступа, дескрипторы графики, а также переменные рабочей области.

Например, в файл startup.m можно ввести строку, которая добавит каталог /home/me/mytools к установленному по умолчанию списку путей доступа addpath /home/me/mytools.

Использование памяти

Система MATLAB требует для хранения каждой матрицы непрерывной области памяти. В частности, образы и анимация могут потреблять очень большие объемы памяти. В дополнение к памяти для хранения матрицы карта пикселей, используемая для образов, требует памяти, пропорциональной площади изображения. Так, например, изображение 500×500 цветных пикселей требует 2 Мбайт оперативной памяти. Если требуется 10 изображений такого размера, то уже необходимо 20 Мбайт, что является очень большим объемом. Чтобы уменьшить объем памяти, требуемый для этих операций, надо ограничить размер выводимых изображений.

Разрешение проблем выделения памяти. Если отсутствует фрагмент памяти, достаточный для размещения матрицы, то возникает ошибка out of memory, хотя общий объем свободной памяти может быть большим. Это связано с фрагментированием памяти в процессе ее выделения. Чтобы ликвидировать фрагментацию, следует воспользоваться командой pack; другой способ - разместить массивы больших размеров в оперативной памяти заранее в начале сеанса работы.

Управление динамической памятью. Система MATLAB использует для выделения динамической памяти стандартные функции malloc и free языка С. Эти утилиты поддерживают пул памяти, которая распределяется операционной системой в относительно медленном темпе; в свою очередь, для системы MATLAB эта память выделяется намного быстрее. Если пул недостаточен, то утилита malloc запрашивает операционную систему относительно выделения другого фрагмента оперативной памяти, чтобы пополнить пул.

По мере выделения памяти пул может становиться очень большим. Чтобы поддержать быстродействие, утилиты malloc и free не возвращают использованную память операционной системе. Эти подпрограммы исходят из предположения, что если большой объем памяти потребовался один раз,

то в нем возникнет необходимость снова. Побочный эффект этого алгоритма состоит в том, что если MATLAB использовал некоторый объем памяти один раз, то она более недоступна другим программам, даже если MATLAB не использует это. Память пула возвращается операционной системе только по завершении работы системы MATLAB.

Интерактивный доступ к справочной информации и документации

Существуют следующие способы получить информацию о функциях системы MATLAB в процессе сеанса работы:

- команда help;
- команда lookfor;
- меню Help;
- просмотр и вывод на печать страниц документации;
- обращение к Web-серверу фирмы The MathWorks.

Команда Help

Основной и наиболее быстрый способ выяснить синтаксис и особенности применения М-функции - это использовать команду help *<имя М-функции>*. Соответствующая информация появляется непосредственно в командном окне.

Например, команда help magic выведет в командное окно следующую информацию на английском языке:

```
help magic
 MAGIC  Magic square.
  MAGIC(N) is an N-by-N matrix constructed from the integers
  1 through N^2 with equal row, column, and diagonal sums.
  Produces valid magic squares for N = 1,3,4,5,...
```

```
MAGIC  Магический квадрат.
 MAGIC(N) - это матрица размера N×N, построенная из целых чисел
 от 1 до N^2 так, что суммы элементов по строкам, столбцам
 и диагоналям совпадают.
 Формирует правильные магические квадраты для N = 1,3,4,5,...
```

Следует обратить внимание, что текст интерактивной справки использует верхний регистр для написания имен функций и переменных, чтобы выделить их из остальной части текста. Однако при вводе имен функций в командной строке всегда используются символы нижнего регистра, поскольку система MATLAB чувствительна к выбору регистра, а действительные имена функций записываются строчными буквами.

Все функции системы MATLAB, а их более 800, организованы в логические группы, и структура каталогов основана на этой организации. Например, все функции линейной алгебры находятся в каталоге matfun. Можно распечатать все функции этого каталога с короткими пояснениями, если использовать команду help matfun:

```
help matfun
  Matrix functions - numerical linear algebra.

  Matrix analysis.
    norm          - Matrix or vector norm.
    normest       - Estimate the matrix 2-norm.

  Linear equations.
    \ and /       - Linear equation solution; use "help slash".
    inv           - Matrix inverse.

  Eigenvalues and singular values.
    eig           - Eigenvalues and eigenvectors.
    svd           - Singular value decomposition.

  Matrix functions.
    expm          - Matrix exponential.
    logm          - Matrix logarithm.

  Factorization utilities
    qrdelete      - Delete column from QR factorization.
    qrinsert      - Insert column in QR factorization.
```

Команда help сама по себе выводит на экран список каталогов

```
help
HELP topics:

matlab\general        -  General purpose commands.
matlab\ops            -  Operators and special characters.
matlab\lang           -  Programming language constructs.
matlab\elmat    -  Elementary matrices and matrix manipulation.
```

Команда lookfor

Эта команда позволяет выполнить поиск М-функции по ключевому слову; при этом анализируется первая строка комментария, и она же выводится на экран, если в ней встретилось ключевое слово. Например, в системе MATLAB нет М-функции с именем inverse и поэтому на команду help inverse ответом будет - inverse.m not found.

Однако команда lookfor inverse найдет не менее дюжины совпадений, и это будет зависеть от того, какие ППП подключены к системе MATLAB.

```
lookfor inverse
INVHILB Inverse Hilbert matrix.
ACOS    Inverse cosine.
ACOSH   Inverse hyperbolic cosine.

IFFTN N-dimensional inverse discrete Fourier transform.
IPERMUTE Inverse permute array dimensions.
ICCEPS Inverse complex cepstrum.
IDCT Inverse discrete cosine transform.
idctold.m: %IDCT Inverse discrete cosine transform.
```

Добавление опции -all команде lookfor в форме lookfor <*шаблон*> -all позволяет просматривать все строки комментария к М-функции, а не только первую строку.

Меню Help

Это меню командного окна системы MATLAB показано на рис. 1.10 и позволяет активизировать следующие окна:

Help Window	Окно справки
Help Tips	Окно справки для получения подсказки
Help Desk (HTML)	Доступ к справочным системам на жестком диске или CD-ROM
Examples and Demos	Окно демонстрационной подсистемы
About MATLAB	Информация об установленной версии
Subscibe (HTML)	Подписка на услуги фирмы The MathWorks, Inc

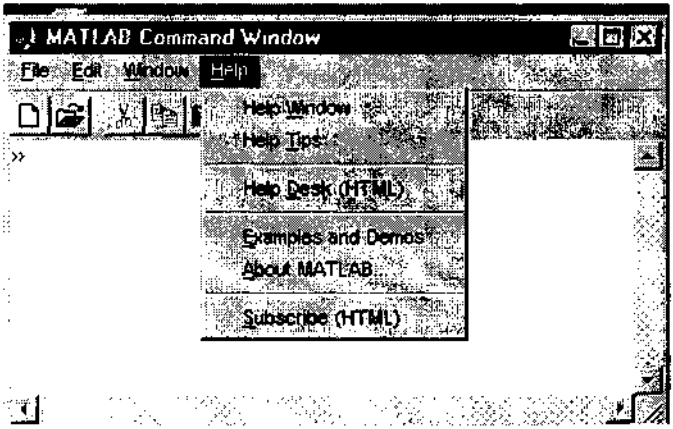


Рис. 1.10

Окно справки Help Window. Это окно может быть вызвано несколькими способами: как опция Help Window меню Help, нажатием кнопки ? инструментальной панели либо с помощью команды helpwin (рис. 1.11).

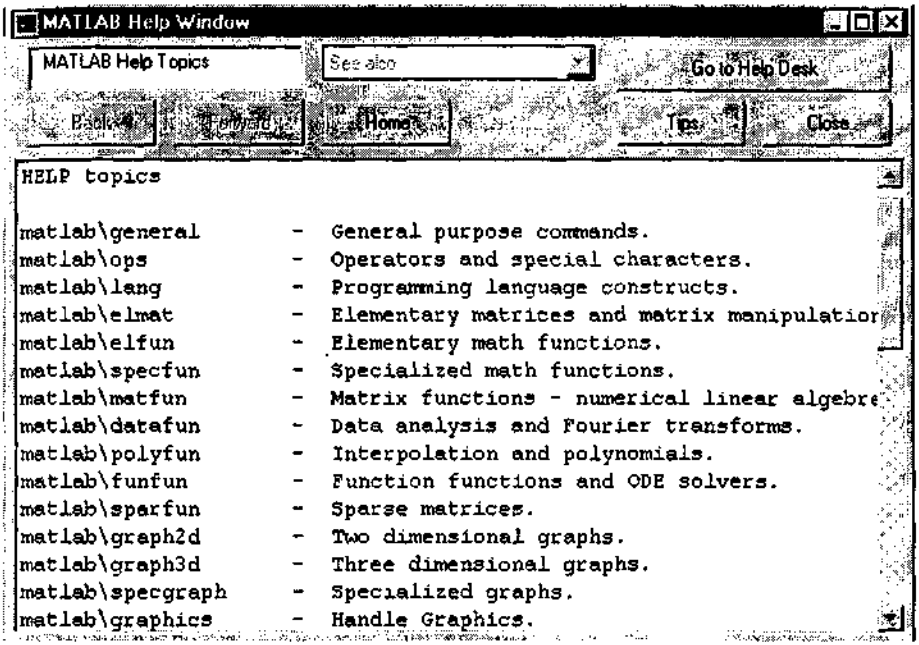


Рис. 1.11

На этом рисунке показано начальное окно MATLAB Help topics; выделяя любую из строк списка и дважды щелкая левой кнопкой мыши, можно переходить к спискам соответствующих разделов. Такое действие аналогично команде helpwin *<имя раздела>*.

Ниспадающий список See also в этом случае неактивен. Кнопки Back, Forward, Home позволяют переходить от одного активизированного окна к другому либо возвратиться к начальному окну.

В правом верхнем углу окна расположены 3 кнопки Go to Help Desk, Tips, Close.

Кнопка Tips выводит окно подсказок (рис. 1.12). В этом окне ниспадающее меню активизировано и позволяет выполнить ряд дополнительных справочных команд More help info help (HTML), lookfor, which, demo, general.

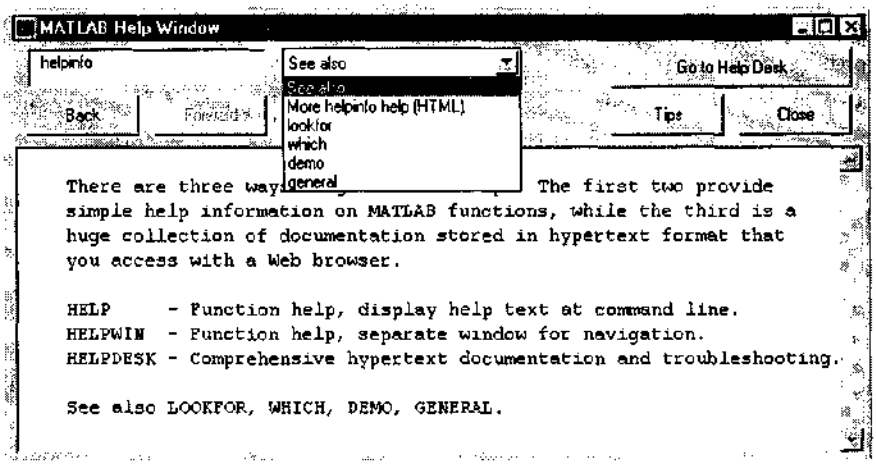


Рис. 1.12

Опция Help Desk. Эта опция позволяет получить доступ к большому объему справочной информации и к документации по системе, размещаемой либо на жестких дисках, либо на диске CD-ROM в рамках используемого персонального компьютера.

Многие из документов используют язык гипертекстовых ссылок HTML (HyperText Markup Language) и доступны для просмотра с помощью таких средств, как Netscape Navigator или Microsoft Explorer.

Эта опция может быть также инициирована с помощью команды **helpdesk**.

Все операторы и функции системы MATLAB описаны в формате HTML и содержат больше подробностей и примеров, чем справки по команде **help**. Доступны HTML версии разных документов, включая описания, руководства пользователя по системе и пакетам прикладных программ. Реализованная поисковая система позволяет выполнить необходимые запросы.

Опция About MATLAB. Эта опция выводит на экран заставку системы с указанием версии и принадлежности пользователю (рис. 1.13).

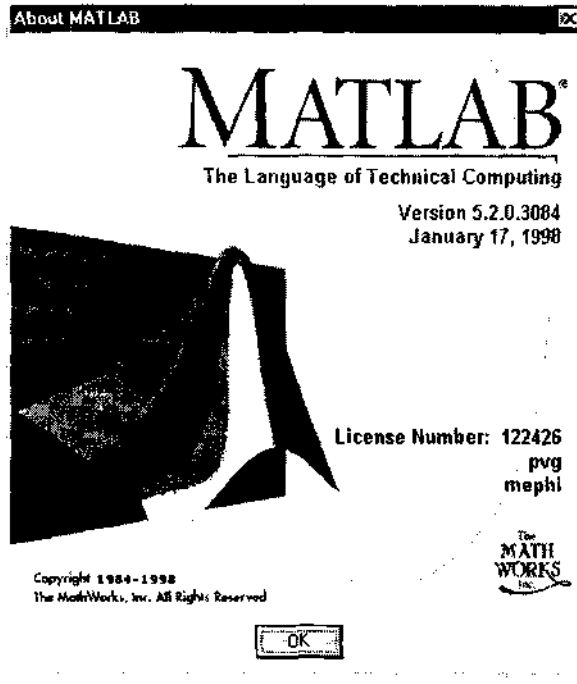


Рис. 1.13

Просмотр и распечатка документации. Версии справочной документации доступны для просмотра и распечатки в формате PDF (Portable Document Format) с помощью средства Adobe's Acrobat. Оно позволяет просматривать текст в формате печатной страницы, с полным набором шрифтов, графики и изображений, с полным ощущением чтения книги. Одновременно это и наилучший способ получения копий нужных страниц.

Web-сервер фирмы MathWorks. Если ваш компьютер подсоединен к сети Internet, то можно реализовать через механизм Help Desk соединение с WWW-сервером фирмы-производителя и выйти на страницу системы MATLAB. Можно также воспользоваться электронной почтой, чтобы задать вопросы, внести предложения или сообщить об обнаруженных ошибках. Можно воспользоваться поисковой системой WWW-сервера, чтобы сделать запрос к постоянно обновляемой базе данных технической поддержки. Возможности WWW-сервера постоянно расширяются, и вы можете более подробно ознакомиться с его информационными возможностями, непосредственно связавшись с его зеркалом в Европе по адресу www-europe.mathworks.com.

2. СПРАВОЧНЫЕ И УПРАВЛЯЮЩИЕ КОМАНДЫ И ФУНКЦИИ

Начнем изучение системы MATLAB с команд и функций общего назначения, в состав которых входят справочные команды, команды управления рабочей областью памяти, путями доступа, командным окном и некоторые команды операционной системы, доступные из среды системы MATLAB.

Справочные команды

DEMO — **Демонстрация возможностей системы MATLAB**

Синтаксис:

demo
demo matlab | toolbox | simulink | blockset | stateflow
demo matlab <*имя_раздела*>
demo toolbox <*имя_ППП*>

Описание:

Команда demo открывает графическое окно MATLAB Demo Window, в котором содержится перечень всех подсистем и пакетов прикладных программ, инсталлированных на данном компьютере, а также краткое описание и перечень демонстраций для этих подсистем и пакетов.

Команда demo matlab | toolbox | simulink | blockset | stateflow открывает графическое окно MATLAB Demo Window и выделяет строку, соответствующую одной из этих подсистем, а также выводит ее краткое описание и перечень демонстраций.

Команда demo matlab <*имя_раздела*> открывает графическое окно MATLAB Demo Window и выделяет строку, соответствующую данному разделу системы MATLAB, выводит его краткое описание и перечень демонстраций.

Команда demo toolbox <*имя_ППП*> открывает графическое окно MATLAB Demo Window и выделяет строку, соответствующую данному пакету прикладных программ (ППП), выводит его краткое описание и перечень демонстраций.

Сопутствующие команды: HELP, HELPDESK, HELPWIN, TOUR.

INFO — ***Информация о системе MATLAB***

Синтаксис:

info
info <*путь_доступа_к_ППП*>

Описание:

Команда info выводит на экран информацию о системе MATLAB и фирме The MathWorks, Inc. - производителе этой системы.

Команда info <*путь_доступа_к_ППП*> выводит на экран файл Readme для указанного ППП.

Пример:

Команда info позволяет вывести следующую информацию о системе MATLAB (приведены переводы некоторых фрагментов):

Система MATLAB доступна на следующих платформах: PC, Macintosh, Sun, VMS, HP, SGI, LINUX, RS/6000.

Продукты фирмы The MathWorks (в алфавитном порядке):

MATLAB
Simulink

Applix Link
Communications Toolbox
Control System Toolbox
DSP Blockset
Excel Link
Financial Toolbox
Fixed-Point Blockset
Frequency Domain System Identification Toolbox
Fuzzy Logic Toolbox
Higher-Order Spectral Analysis (Hi-Spec) Toolbox
Image Processing Toolbox
LMI Control Toolbox
Mapping Toolbox
MATLAB Compiler
MATLAB C Math Library
MATLAB C++ Math Library
Model Predictive Control Toolbox
μ-Analysis and Synthesis Toolbox
NAG Foundation Toolbox
Neural Network Toolbox
Nonlinear Control Design Blockset
Optimization Toolbox
Partial Differential Equation Toolbox
QFT Control Design Toolbox
Real-Time Workshop
Robust Control Toolbox
RTW Ada Extension
Signal Processing Toolbox
Simulink Accelerator
Spline Toolbox
Stateflow™
Stateflow Coder
Statistics Toolbox
Symbolic Math and Extended Symbolic Math Toolboxes
System Identification Toolbox
Wavelet Toolbox
Chemometrics Toolbox
An Introduction to MATLAB video

Если вы являетесь официальным пользователем системы MATLAB, то можете зарегистрироваться и стать клиентом услуги Access, которая позволяет бесплатно получать:

- техническую поддержку и обслуживание (только в США);
- предварительное уведомление о выпуске нового продукта (через электронную почту);

- пароль для доступа к серверу Access с целью проверить:
  - состояние заказа,
  - персональный пароль лицензии,
  - место установки продукта и другую лицензионную информацию,
  - информацию о пользователе;
- доступ к документации в режиме online;
- бесплатную подписку на информационные бюллетени:
  - MATLAB News & Notes,
  - MATLAB Digest.

Чтобы узнать больше об услуге Access или стать ее клиентом, следует посетить сервер фирмы MathWorks в сети Internet по адресу www.mathworks.com или его зеркало в Европе www-europe.mathworks.com.

Чтобы стать клиентом Access, сообщите ваше имя, место работы, почтовый адрес, адрес электронной почты, телефон и лицензионный номер, который можно узнать с помощью команды ver. Вышлите эту информацию по электронной или факсимильной почте по адресу:

The MathWorks, Inc.
24 Prime Park Way
Natick, MA 01760-1500 USA
Fax: +508-647-7101
E-mail: access@mathworks.com
Web: www.mathworks.com

Сопутствующие команды: WHATSNEW.

WHATSNEW **Вывод на экран файлов readme системы MATLAB и ППП**

Синтаксис:

whatsnew <*имя_подсистемы или ППП*>

Описание:

Команда whatsnew<*имя_подсистемы или ППП*> выводит на экран файл Readme для данной подсистемы или ППП. Файл Readme, если он существует для данного продукта, содержит описание новых функциональных возможностей, которые не описаны в документации по пакету или подсистеме.

Пример:

Вывод на экран файла Readme системы MATLAB:

whatsnew matlab

Ниже приведен в переводе фрагмент файла Readme, относящийся к новым возможностям версии MATLAB 5.2:

Файл Readme для системы MATLAB.
18-Dec-1997
==========
Что нового в MATLAB 5.2
==========

Изменения в языке

В версии MATLAB 5.2 добавлено две новых языковых конструкции:

try, ..., catch, ..., end - для упрощения написания процедур обработки ошибок;

persistent <*имя_переменной*> - для поддержки работы с глобальными переменными внутри М-функций.

Оба нововведения направлены на то, чтобы упростить создание надежных приложений.

Улучшена реализация и расширены возможности некоторых М-функций:

- М-функции lower, upper, strcmp, strrep, strncmp стали работать быстрее;
- М-функция load теперь имеет выходной аргумент;
- М-функции subsref и subsasgn применимы для всех типов массивов;
- для многомерных массивов пустой массив [ ] равносилен функциям CAT(1,...) или CAT(2,...);
- ускорено исполнение М-функций fft, fft2, fftn;
- М-функции обработки множеств типа unique, intersect и т. п. могут применяться к массивам ячеек, содержащих строки;
- для корневого объекта введено новое свойство, которое позволяет запретить запись стека в запоминающее устройство:

set(0, 'recursionLimit', Nframes);

- расширены возможности М-функции hdf для поддержки интерфейсов SD, VS, V, AN;
- М-функции интерполяции типа interp1, interp2 и т. п. теперь реализуют метод интерполяции сплайнами.

Новые М-функции:

lastwarn	- получить текст последнего предупреждения;
mkdir	- создать каталог или папку;
gsvd	- вычислить обобщенное SVD-разложение;
strcmpi	- сравнение строк без учета различия строчных и прописных;
actxcontrol	- поддержка объектов при работе с протоколом ActiveX.

Дескрипторная графика

- новые функции управления подсветкой и положением съемочной камеры (раздел GRAPH3D);

- добавлено новое значение opengl для свойства renderer графического объекта figure: set(gcf, 'renderer', 'opengl');
- управление размещением изображений на графических кнопках;
- добавлен новый графический объект contextmenu;
- пояснения к объектам управления *GUI*;
- бета-версия графического редактора PLOTEDIT.

Новые функции (по разделам):

datafun:	ifftshift;
datatypes:	substruct;
funfun:	ntrp23t, ntrp23tb, ode23t, ode23tb;
general:	copyfile, maedispsubarray, maesize, matlabpath, mauifunc, mkdir, regedit;
graph2d:	dokeypress, domymenu, doresize, enddrag, getobj, makedraggable, middrag, plotedit, prepdrag, putdowntext, scribeaxesdlg, scribetoolbar;
graph3d:	camdolly, camlight, camlookat, camorbit, campan, campos, camproj, camroll, camtarget, camup, camva, camzoom, daspect, lightangle, objbounds, pbaspect, xlim, ylim, zlim;
graphics:	handle2struct, struct2handle, uicontextmenu;
iofun:	hdfan, hdfdf24, hdfdfr8, hdfh, hdfhd, hdfhe, hdfml, hdfsd, hdfv, hdfvf, hdfvh, hdfvs, hgload, hgsave;
lang:	catch, lastwarn, mislocked, mlock, munlock, persistent, try;
matfun:	cholupdate, gsvd, qrupdate;
polyfun:	mpoles, splncore;
strfun:	strcmpi, strncmpi;
uitools:	uirestore, uisuspend;
winfun:	actxcontrol, actxserver, mwsamp, sampev.

Удалены М-функции: whichcls, maeasgn

Сопутствующие команды: VER, HELP, LOOKFOR, README.

SUBSCRIBE **Подписка на информационный бюллетень**

Синтаксис:

subscribe

Описание:

Команда subscribe позволяет пользователю стать подписчиком информационного бюллетеня MathWorks Newsletter. При этом подписчики получают и другую информацию о последних разработках системы MATLAB. Для того чтобы стать подписчиком, не надо быть зарегистрированным пользователем системы, и, кроме того, подписка является бесплатной.

Подписчики смогут получать бюллетень MathWorks Newsletter регулярной почтой, а электронный бюллетень MATLAB News Digest - по электронной почте.

По команде subscribe вам будет предложено указать вашу фамилию и адрес, и если вы находитесь в среде ОС UNIX, то будет сформировано email-сообщение на фирму The MathWorks, Inc. Если вы находитесь в другой операционной среде, то будет создан регистрационный файл, который можно распечатать и направить по факсу или почтой в адрес фирмы.

Сопутствующие команды: HTML-справка.

HOSTID **Определение идентификационного номера сервера**

Синтаксис:

hostid

Описание:

Команда hostid в среде Windows возвращает массив, состоящий из одной ячейки, которая содержит строку с регистрационным номером пользователя.

Пример:

```
hostid
  '122426'
```

Примечание:

Не рекомендуется использовать при написании программ, поскольку в последующих версиях эта функция будет отсутствовать.

Сопутствующие команды: VER.

LICENSE **Определение номера лицензии**

Синтаксис:

license

Описание:

Команда license возвращает строку с лицензионным номером используемой версии системы MATLAB, которая совпадает с регистрационным номером пользователя.

Пример:

```
license
ans = 122426
```

Сопутствующие команды: HOSTID, VER.

VER **Используемая версия системы MATLAB и ППП**

Синтаксис:

ver

ver <*имя_ППП*>

Описание:

Команда ver выводит на экран номера текущих версий системы MATLAB и ППП.

Команда ver <*имя_ППП*> выводит на экран информацию о текущей версии запрошенного ППП.

Пример:

```
ver
-----------------------------------------------
MATLAB Version 5.2.0.3084 on PCWIN
MATLAB License Identification Number: 122426
-----------------------------------------------
MATLAB Toolbox                    Version 5.2      18-Dec-1997
```

В данном случае в состав инсталлированной системы входит только система MATLAB.

Сопутствующие команды: HOSTID, VERSION.

VERSION **Используемая версия системы MATLAB**

Синтаксис:

```
version
[v, d] = version
```

Описание:

Команда version возвращает строку с указанием используемой версии системы.

Функция [v, d] = version возвращает не только номер версии, но и дату ее выпуска.

Пример:

```
version
ans =  5.2.0.3084
[V, D] = version
V = 5.2.0.3084
D = Jan 17 1998
```

Примечание:

Не рекомендуется использовать при написании программ, поскольку в последующих версиях эта функция будет отсутствовать.

Сопутствующие команды: VER.

HELPINFO **Информация о справочных подсистемах системы MATLAB**

Синтаксис:

helpinfo

Описание:

Команда helpinfo возвращает следующую информацию о справочных подсистемах системы MATLAB.

Существует 3 способа получить интерактивную справку. Первые два связаны с простейшей справочной информацией относительно функций системы, третий - это огромная коллекция документации, сохраненной в формате гипертекста и которую можно просмотреть с помощью Web-браузера.

Для этого предназначены следующие команды:

Команда	*Назначение*
help	Вывод справки в командной строке
helpwin	Вывод справки в отдельном графическом окне
helpdesk	Исчерпывающая информация в HTML-формате

Сопутствующие команды: DEMO, GENERAL, LOOKFOR, WHICH.

HELP **Справка о командах и функциях системы**

Синтаксис:

help
help <*раздел/функция*>
help <*специальный символ*>

Описание:

Команда help выводит на экран список разделов системы MATLAB.

Список разделов системы MATLAB:

matlab\general	Команды общего назначения
matlab\ops	Операторы и специальные символы
matlab\lang	Конструкции языка и функции отладки
matlab\elmat	Матрицы и операции над ними
matlab\elfun	Элементарные математические функции
matlab\specfun	Специальные математические функции
matlab\matfun	Линейная алгебра
matlab\datafun	Анализ данных и преобразование Фурье
matlab\polyfun	Полиномы и интерполяция
matlab\funfun	Нелинейные численные методы и решатели ОДУ

matlab\sparfun	Разреженные матрицы
matlab\graph2d	Двумерная графика
matlab\graph3d	Трехмерная графика
matlab\specgraph	Специальная графика
matlab\graphics	Дескрипторная графика
matlab\uitools	Графический интерфейс пользователя GUI
matlab\strfun	Функции обработки символьных строк
matlab\iofun	Функции ввода/вывода
matlab\timefun	Функции времени и дат
matlab\datatypes	Типы данных и структуры
matlab\winfun	Интерфейс с ОС Windows (DDE /ActiveX)
matlab\demos	Демонстрации и примеры
toolbox\local	Выбор характеристик

Команда help <*раздел/функция*> выводит перечень функций или описание самой функции. При наборе команды help не следует указывать полный путь доступа к разделу или команде, а достаточно указать лишь имя раздела или функции.

Пример:

Команды help general, help matlab/general выводят на терминал один и тот же листинг каталога toolbox/matlab/general.

Команда help <*специальный символ*> выводит список специальных символов.

Сопутствующие команды: LOOKFOR, WHAT, WHICH, DIR, MORE.

HELPWIN **Подсистема интерактивной справки**

Синтаксис:

helpwin

helpwin <*раздел*> | <*функция*>

Описание:

Команда helpwin открывает графическое окно, которое позволяет реализовать интерактивную навигацию по справочной системе.

Команда helpwin <*раздел*> | <*функция*> открывает графическое окно справки, в котором выводится текст справки по данному разделу или функции; кроме того, устанавливаются связи с теми функциями и командами, которые указаны в разделе 'See Also' (*Сопутствующие команды, функции, операторы, переменные*).

Пример:

helpwin helpwin

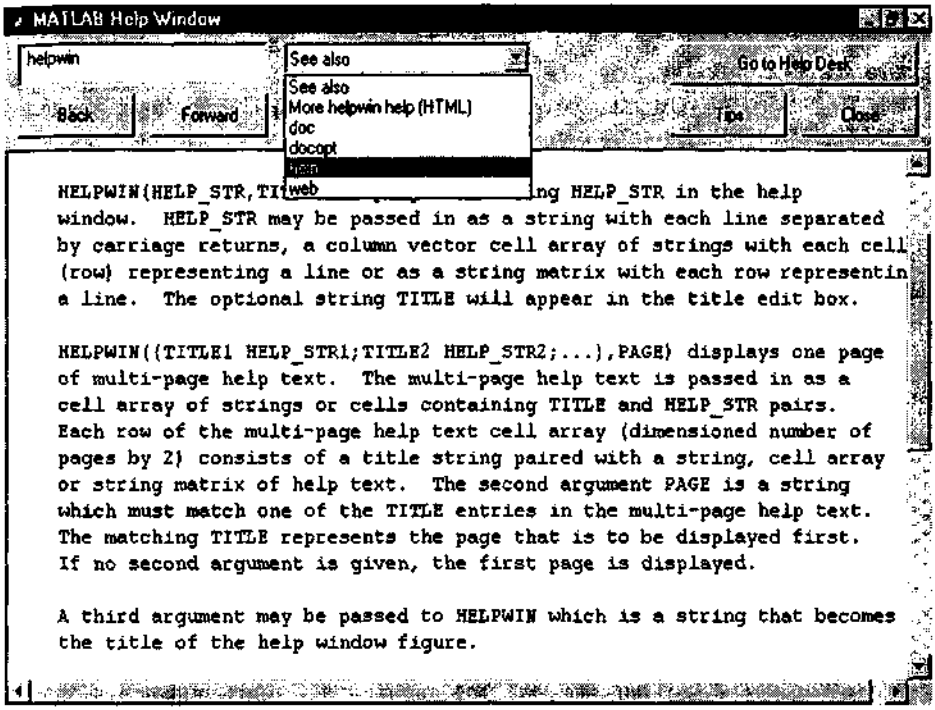


Рис. 2.1

Сопутствующие команды: DOC, DOCOPT, HELP, WEB.

HELPDESK **Доступ к HTML-документации**

Синтаксис:

helpdesk

Описание:

Команда helpdesk загружает главную страницу подсистемы MATLAB Help Desk в Web-браузер.

Сопутствующие команды: DOC, DOCOPT, HELP, HELPWIN, WEB.

LOOKFOR **Поиск по ключевому слову**

Синтаксис:

lookfor <*ключевое_слово*>

lookfor <*ключевое_слово*> -all

Описание:

Команда lookfor <*ключевое_слово*> отыскивает заданную в виде ключевого слова строку символов в первой строке комментария, просматривая все М-файлы из списка путей доступа MATLABPATH. Из всех файлов, в которых будет обнаружено совпадение с ключевым словом, команда lookfor извлечет первую строку. Команда more on позволяет приостановить вывод на экран текста справки, если он занимает несколько экранов.

Команда lookfor <*ключевое_слово*> -all ищет заданную строку символов во всем блоке комментария, просматривая все М-файлы из списка путей доступа MATLABPATH.

Пример:

Команда lookfor inverse выводит по меньшей мере два десятка строк, содержащих ключевое слово inverse.

Сопутствующие команды: DIR, HELP, WHO, WHAT, WHICH.

WHAT **Вывод списка файлов**

Синтаксис:

what <*без_параметров*>

what <*имя_каталога*>

Описание:

Команда what <*без параметров*> выводит списки файлов текущего каталога в виде массива структур со следующими полями:

path - путь доступа к текущему каталогу;
m - массив ячеек для имен М-файлов;
mat - массив ячеек для имен MAT-файлов;
mex - массив ячеек для имен MEX-файлов;
mdl - массив ячеек для имен MDL-файлов;
p - массив ячеек для имен P-файлов;
classes - массив ячеек для имен классов.

Команда what <*имя_каталога*> выводит список файлов указанного каталога, который должен быть описан в переменной MATLABPATH. При наборе команд не следует указывать полный путь доступа, а достаточно указать лишь последние элементы этого пути.

Пример:

Команды what general, what matlab/general выводят один и тот же список файлов каталога toolbox/matlab/general.

```
what general
М-файлы в каталоге E:\MLAB52\toolbox\matlab\general
```

Contents	docroot	maedispsubarray	prepender
Readme	dos	maeresize	profile
addpath	echo	maesize	profsumm
binpatch	edit	matlabpath	pwd
cd	editpath	mauifindexe	quit
clear	exit	mauifunc	regedit
computer	format	mdbstatus	rmpath
copyfile	genpath	memory	save
dbclear	getenv	mex	subscribe
dbcont	help	mexdebug	type
dbdown	helpdesk	miedit	unix
dbmex	helpinfo	miolereg	ver
dbquit	helpwin	miport	vms
dbstack	info	mkdir	web
dbstatus	inmem	more	what
dbstep	isieee	nnload	whatsnew
dbstop	isppc	notebook	which
dbtype	isstudent	openvar	who
dbup	isunix	pack	whos
debug	isvms	path	workspace
diary	load	path2rc	
dir	lookfor	pathtool	
doc	ls	pcode	

MEX-файлы в каталоге E:\MLAB52\toolbox\matlab\general

find_netscape	ibrowse	matver	simver
getprofl	iebrowse	rtwver	wrtprofl

P-файлы в каталоге E:\MLAB52\toolbox\matlab\general

helpwin

Классы в каталоге E:\MLAB52\toolbox\matlab\general

char

Сопутствующие команды: DIR, WHO, WHICH, LOOKFOR.

WHICH **Справка о пути доступа к функции или файлу**

Синтаксис:

which <*имя_функции, файла или переменной*>
which <*имя функции или файла*> -all
which <*имя_файла.**>
which <*имя_функции 1*> in <*имя_функции 2*>
which <*имя_функции*>(arg1, arg2, ...)
S = which(...)
W = which(..., '-all')

Описание:

Команда which <*имя_функции, файла или переменной*> выводит на экран полное описание пути доступа к данной функции, файлу или переменной.

Это могут быть М-, MAT-, MEX-, P-файлы, графические функции подсистемы SIMULINK, встроенные функции системы MATLAB или переменные рабочей области.

Команда which <*имя_функции или файла*> -all выводит на экран все пути доступа к функции или файлу с данным именем. Первым в списке, как правило, будет тот путь, который выводится по команде без опции -all. Другие пути доступа являются скрытыми или могут быть использованы при особых обстоятельствах. Опция -all может быть применена для всех форм команды which.

Команда which <*имя_файла.**> выводит на экран путь доступа к файлу с указанным расширением.

Команда which <*имя_функции 1*> in <*имя_функции 2*> выводит на экран путь доступа к М-файлу с именем <*имя_функции 1*>, который вызывается из М-файла с именем <*имя_функции 2*>. Эта команда полезна, когда необходимо выявить подпрограмму или частную версию функции, используемую в модуле. Если выполняется отладка М-файла (модуля) с именем <*имя_функции 2*>, то форма which <*имя_функции 1*> равносильна описываемой команде.

Команда which <*имя_функции*>(arg1, arg2, ...) выводит на экран путь доступа к М-функции с указанным именем и заданными входными аргументами.

Функция S = which(...) возвращает строковую переменную S, которая содержит информацию о пути доступа к заданной функции, файлу или переменной.

Функция W = which(..., '-all') возвращает строковую переменную W, которая содержит информацию о всех возможных путях доступа к заданной функции или файлу.

Пример:

```
which inv
    inv is a built-in function.
which pinv
E:\MLAB52\toolbox\matlab\matfun\pinv.m
g = inline('sin(x)'),
which feval(g)
inline/feval.m
```

Первая команда сообщает, что inv - встроенная функция, вторая - что М-файл pinv находится в каталоге E:\MLAB52\toolbox\matlab\matfun, третья - путь доступа inline/feval.m.

Если запрашиваемый файл, функция или переменная отсутствуют, то будет получен ответ

<*имя_функции, файла или переменной*> cannot be found
<*имя_функции, файла или переменной*> *не найдено.*

Сопутствующие команды: DIR, HELP, WHO, WHAT, EXIST, LOOKFOR.

Характеристики операционной среды системы MATLAB

MATLABRC **Главный файл запуска системы MATLAB**

Синтаксис:

matlabrc

Описание:

Команда matlabrc в момент запуска системы MATLAB автоматически выполняет главный М-файл запуска matlabrc.m и пользовательский файл startup.m, если он существует. В многопользовательской или сетевой среде файл matlabrc.m резервируется для системного менеджера. В однопользовательской среде файл matlabrc.m применяется для задания стандартных характеристик экрана терминала, который является корневым объектом (root object) системы MATLAB. Ему присваивается значение дескриптора 0; это единственный графический объект, который не имеет родителей, но сам он порождает графический объект figure с дескриптором 1 (номер фигуры).

Корневой объект имеет следующие характеристики:

Характеристики и допустимые значения	*Текущие значения*
	CallbackObject = [ ]
Language	Language = russian
CurrentFigure	CurrentFigure = [ ]
Diary: [on | off]	Diary = off
DiaryFile	DiaryFile = diary
Echo: [on | off]	Echo = off
ErrorMessage	ErrorMessage =
Format: [short | long | shortE | longE | shortG | longG | hex | bank | + | rational]	Format = short
FormatSpacing: [loose | compact]	FormatSpacing = loose
PointerLocation	PointerLocation = [634 45]
	PointerWindow = [0]
Profile: [on | off]	Profile = off
ProfileFile	ProfileFile =
ProfileCount	ProfileCount = [ ]
ProfileInterval	ProfileInterval = [0.01]
RecursionLimit	RecursionLimit = [500]
ScreenDepth	ScreenDepth = [16]
	ScreenSize = [1 1 640 480]
ShowHiddenHandles: [on | {off}]	ShowHiddenHandles = off
Units: [inches | centimeters | normalized | points | pixels | characters]	Units = pixels
AutomaticFileUpdates: [on | off]	AutomaticFileUpdates = on
ButtonDownFcn	ButtonDownFcn =

Children	Children = [ ]
Clipping: [{on} | off]	Clipping = on
CreateFcn	CreateFcn =
DeleteFcn	DeleteFcn =
BusyAction: [{queue} | cancel]	BusyAction = queue
HandleVisibility: [{on} | callback | off]	HandleVisibility = on
HitTest: [{on} | off]	HitTest = on
Interruptible: [{on} | off]	Interruptible = on
Parent	Parent = [ ]
Selected: [on | off]	Selected = off
SelectionHighlight: [{on} | off]	SelectionHighlight = on
Tag	Tag =
	Type = root
UIContextMenu	UIContextMenu = [ ]
UserData	UserData = [ ]
Visible: [{on} | off]	Visible = on

В однопользовательской среде пользователь может также создать файл startup.m, в котором могут быть описаны индивидуальные пути доступа, применяемые физические константы, коэффициенты перевода единиц измерения и другие параметры, необходимые для создания уникальной операционной среды на рабочем месте пользователя.

Алгоритм:

М-файл matlabrc.m описывает пути доступа к компонентам системы MATLAB, содержит установки штатных параметров экрана:

```
set(0, 'defaultuicontrolbackgroundcolor', gray);
set(0, 'defaultfigureposition', rect);,
```

а также условный оператор

```
if (exist('startup.m') == 2)
startup
end
```

Сопутствующие команды: QUIT, !, PATH.

STARTUP **Пользовательский файл запуска системы MATLAB**

Синтаксис:

```
startup
```

Описание:

Команда startup в момент запуска системы MATLAB используется главным файлом запуска matlabrc.m для установления пользовательских параметров операционной среды. Если М-файл отсутствует, то устанавливаются стандартные параметры среды, задаваемые файлом matlabrc.m.

Сопутствующие команды: MATLABRC.

MATLABPATH **Путь доступа, используемый по умолчанию**

Синтаксис:

```
matlabpath
matlabpath(p)
p = matlabpath
```

Описание:

Команда matlabpath позволяет вывести на экран список путей доступа в удобочитаемой форме. Реализована эта команда как встроенная функция.

Команда matlabpath(p) изменяет прежний список доступа на список p.

Функция p = matlabpath присваивает строковой переменной p список путей доступа, используемый в системе MATLAB.

Разработчики настоятельно рекомендуют применять эту команду только для получения информации, а при работе пользоваться командой path.

Пример:

```
matlabpath
MATLABPATH
E:\MLAB52\toolbox\matlab\general
E:\MLAB52\toolbox\matlab\ops
E:\MLAB52\toolbox\matlab\lang
E:\MLAB52\toolbox\matlab\elmat
E:\MLAB52\toolbox\matlab\elfun
E:\MLAB52\toolbox\matlab\specfun
E:\MLAB52\toolbox\matlab\matfun
E:\MLAB52\toolbox\matlab\datafun
E:\MLAB52\toolbox\matlab\polyfun
E:\MLAB52\toolbox\matlab\funfun
E:\MLAB52\toolbox\matlab\sparfun
E:\MLAB52\toolbox\matlab\graph2d
E:\MLAB52\toolbox\matlab\graph3d
E:\MLAB52\toolbox\matlab\specgraph
E:\MLAB52\toolbox\matlab\graphics
E:\MLAB52\toolbox\matlab\uitools
E:\MLAB52\toolbox\matlab\strfun
E:\MLAB52\toolbox\matlab\iofun
E:\MLAB52\toolbox\matlab\timefun
E:\MLAB52\toolbox\matlab\datatypes
E:\MLAB52\toolbox\matlab\winfun
E:\MLAB52\toolbox\matlab\demos
E:\MLAB52\toolbox\local
```

Сопутствующие команды: PATH, PATHDEF, PATHSEP.

DOCOPT **Параметры для работы с Web-браузером**

Синтаксис:

[doccmd, options, docpath] = docopt

Описание:

М-файл docopt предназначен для того, чтобы пользователь или системный менеджер могли отредактировать его с целью указать, как можно обратиться к программам и файлам, которые разрешают просмотр документации в интерактивном режиме.

Функция [doccmd, options, docpath] = docopt возвращает 3 строковые переменные:

doccmd - это строка, содержащая команду, которая указывает, что для просмотра документации в интерактивном режиме будет применяться команда doc. По умолчанию эта строка для среды Windows имеет неприсвоенное значение;

options - это строка, содержащая дополнительные параметры конфигурации, которые используются вместе со строкой doccmd при вызове команды doc. По умолчанию эта строка для среды Windows имеет неприсвоенное значение;

docpath - это строка, содержащая путь доступа к файлам документации, просматриваемой в интерактивном режиме. Если эта строка является пустой, то поиск файлов по команде doc реализуется автоматически.

Сопутствующие команды: DOC.

DOC **Просмотр HTML-документации в Web-браузере**

Синтаксис:

doc

doc <*имя_функции*>

doc <*имя_ППП*> / <*имя_функции*>

Описание:

Команда doc стартует сеанс работы с подсистемой Help Desk для просмотра документов в стандарте HTML.

Команда doc <*имя_функции*> выводит на экран HTML-документацию по данной функции. Если функция является переопределяемой, то соответствующая информация по переопределяемым функциям также будет выводиться в командном окне.

Команда doc <*имя_ППП*> / <*имя_функции*> выводит на экран HTML-документацию по функции из указанного ППП.

Пример:

doc eig

doc symbolic/eig

Сопутствующие команды: DOCOPT.

PRINTOPT **Задание опций печати по умолчанию**

Синтаксис:

[pcmd, dev] = printopt

Описание:

Функция printopt является М-файлом, который пользователь или менеджер системы могут редактировать, чтобы установить тип принтера и его назначение по умолчанию.

Функция [pcmd, dev] = printopt возвращает две строковые переменные pcmd и dev.

Переменная pcmd - это строка, содержащая инструкцию печати, которую использует команда print, чтобы послать файл на принтер. Для среды Windows штатное значение переменной pcmd = PRINT.

Переменная dev - это строка, содержащая тип устройства печати, который использует команда print. Для среды Windows штатное значение переменной dev = -dwin.

Сопутствующие функции и команды: PRINT.

CEDIT **Установить клавиши в режим редактирования**

Синтаксис:

cedit ('on'\'off')
cedit ('emacs')

Описание:

Команда cedit ('off') отключает режим редактирования.

Команда cedit ('on') включает режим редактирования.

Команда cedit ('emacs') восстанавливает определение клавиш, принятое по умолчанию.

В среде MS Windows для редактирования используются следующие клавиши:

Клавиши	*Назначение*
Ctrl-P	Перейти на предыдущую строку
Ctrl-N	Перейти на следующую строку
Ctrl-B	Перейти на один символ влево
Ctrl-F	Перейти на один символ вправо
Ctrl-L	Перейти на одно слово влево
Ctrl-R	Перейти на одно слово вправо
Ctrl-A	Перейти к началу строки
Ctrl-E	Перейти к концу строки
Ctrl-U	Удалить строку

Ctrl-D	Удалить символ
Ctrl-T	Переключение режимов вставки и замены символа
Ctrl-K	Удалить содержимое строки справа от курсора

При работе в среде MS Windows в дополнение к стандартному определению клавиатуры всегда включена опция emacs.

Примечание:

Не рекомендуется использовать при написании программ, поскольку в последующих версиях эта функция будет отсутствовать.

Сопутствующие команды: HTML-справка.

COLORDEF **Установить параметры цвета экрана**

Синтаксис:

colordef white | black | none

colordef(<*номер_фигуры*>, *опция*)

h = colordef(<*номер_новой_фигуры*>, *опция*)

Описание:

Команда colordef white | black | none управляет цветом графических объектов Axes и Figure.

Опции white и black позволяют задать цвет прорисовки осей координат для объекта Axes. При этом устанавливается цвет фона для объекта Figure в виде оттенка серого и изменяются другие параметры так, чтобы обеспечивалась необходимая контрастность при выводе графиков.

Опция none устанавливает значения параметров цвета, соответствующие значениям для версии MATLAB 4; наиболее существенное различие состоит в том, что фоновый цвет для объекта axes принимает значение none, так что фоновый цвет для объекта axes и цвет фона для объекта figure становятся одинакового белого цвета. В этом случае цвет фона для объекта figure следует установить черным.

Команда colordef(<*номер_фигуры*>, *опция*) устанавливает цвет фона для объекта figure, используя значения параметра option: white, black, none. Перед тем как использовать эту команду, необходимо очистить объект Figure с помощью команды clf.

Функция h = colordef(<*номер_новой_фигуры*>, *опция*) возвращает поддержку для новой фигуры, использующей новые опции цвета. Эта форма команды удобна при проектировании графического интерфейса пользователя GUI, когда необходимо управлять параметрами окружения. В этом случае параметру visible следует присвоить значение off, чтобы предотвратить вывод на экран.

Сопутствующие команды: WHITEBG.

WHITEBG **Установить цвет объекта Figure**

Синтаксис:

whitebg
whitebg(C)
whitebg(<*номер_фигуры*>)

Описание:

Команды whitebg и whitebg(C) управляют цветом текущего активного графического объекта Figure: первая команда устанавливает цвета, принятые по умолчанию, вторая - в соответствии с палитрой C.

Команда whitebg(<*номер_фигуры*>) устанавливает цвета для заданного графического объекта Figure. Эта команда обычно используется для того, чтобы переключаться между черным и белым цветом фона объекта Axes.

Команды группы whitebg дают лучшие результаты в том случае, когда для всех объектов Axes данной фигуры установлен одинаковый фон.

Примечание:

Не рекомендуется использовать при написании программ, поскольку в последующих версиях эта функция будет отсутствовать.

Сопутствующие команды: COLORDEF.

GRAYMON **Установить параметры палитры для черно-белых мониторов**

Синтаксис:

graymon

Описание:

Команда graymon устанавливает цвета для объекта Figure, которые обеспечивают наилучшее изображение при использовании черно-белых мониторов.

Примечание:

Не рекомендуется использовать при написании программ, поскольку в последующих версиях эта функция будет отсутствовать.

Сопутствующие команды: COLORDEF, WHITEBG.

Управляющие команды и функции

После набора команды ее можно редактировать, используя обычным образом клавиши управления курсором (←, →, Home и End), символьные клавиши и клавишу Del для удаления символа под курсором или сочетание клавиш Ctrl+[для удаления всей неправильно набранной строки. Редактирование строки можно выполнять как в режиме вставки символов, так и в режиме замены. Для переключения режимов используется клавиша Ins.

Для выполнения каждой команды необходимо нажать клавишу Enter. После этого команда немедленно обрабатывается и, если не получено сообщение об ошибке, результат выводится на экран.

В системе MATLAB предусмотрен кольцевой буфер для хранения команд. Для выбора команды из этого буфера используются клавиши управления курсором: клавиша ↑ для выбора предыдущей и клавиша ↓ для выбора последующей команды. Для выбора команды с известным именем наберите один или несколько начальных символов этой команды и нажмите клавишу ↑.

TYPE **Вывод на терминал содержимого ASCII-файла**

Синтаксис:

type <*имя.****>

type <*имя*>

Описание:

Команда type <*имя.****> выводит на экран произвольный ASCII-файл с заданным расширением.

Команда type <*имя*> выводит на экран М-файл.

Сопутствующие команды: DBTYPE, PARTIALPATH.

WEB **Открыть Web-браузер**

Синтаксис:

web url

stat = web(...)

Описание:

Команда web url открывает Web-браузер и загружает файл или домашнюю страницу сервера с адресом url (Uniform Resource Locator - универсальный указатель ресурса). Этот указатель может быть представлен в любой форме, которая поддерживается браузером; обычно это либо локальный файл, либо адрес узла в сети Internet.

Функция stat = web(...) возвращает результат выполнения команды web в виде переменной stat, которая принимает следующие значения:

Значение stat	*Описание*
0	Успешное соединение
1	Браузер не найден
2	Браузер найден, но не может быть запущен

Используемый Web-браузер можно зафиксировать либо в окне Preferences меню File, либо в М-файле docopt.

Примеры:

Открыть файл foo.html на собственном браузере
web file:///disk/dir1/dir2/foo.html
Открыть файл foo.html, если он размещен на пути доступа системы MATLAB
web(['file:///' which('foo.html')]);
Загрузить страницу узла фирмы The MathWorks на собственный браузер
web 'http://www. mathworks.com
Послать электронную почту
web mailto:email_address

Сопутствующие команды: DOC, DOCOPT.

EDIT **Запуск редактора/отладчика M-File Editor/Debugger**

Синтаксис:

edit
edit <*имя_М-файла*>
edit <*имя_файла.* ***>
edit @<*имя_класса*>/<*имя_М-файла*> | private/<*имя_М-файла*> |
@<*имя_класса*>/private/<*имя_М-файла*>
edit <*имя_М-файла* 1> in <*имя_М-файла* 2>
edit <*имя_М-функции*>(arg1, arg2, ...)

Описание:

Команда edit запускает редактор/отладчик M-File Editor/Debugger.

Команда edit <*имя_М-файла*> запускает отладчик M-File Editor/Debugger и открывает М-файл с указанным именем.

Команда edit <*имя_файла.* ***>* запускает редактор/отладчик M-File Editor/Debugger и открывает файл с расширением ***.

Команды edit @<*имя_класса*>/<*имя_М-файла*> | private/<*имя_М-файла*> | @<*имя_класса*>/private/<*имя_М-файла*> запускает редактор/отладчик M-File Editor/Debugger и открывают М-файлы с указанным именем из каталогов класса и частных каталогов.

Команда edit <*имя_М-файла* 1> in <*имя_М-файла* 2> запускает редактор/отладчик M-File Editor/Debugger и открывает М-файл с именем <*имя_функции 1*>, который вызывается из М-файла с именем <*имя_функции 2*>. Эта команда полезна, когда необходимо отредактировать подпрограмму или частную версию функции, используемую в модуле.

Команда edit <*имя_М-функции*>(arg1, arg2, ...) запускает редактор/отладчик M-File Editor/Debugger и открывает М-функцию с указанным именем и заданными входными аргументами.

Пример:

Используя команду edit feval(g), где g = inline('sin(x)'), открыть файл inline/feval.m. (рис. 2.2)

```
g = inline('sin(x)')
edit feval(g)
```

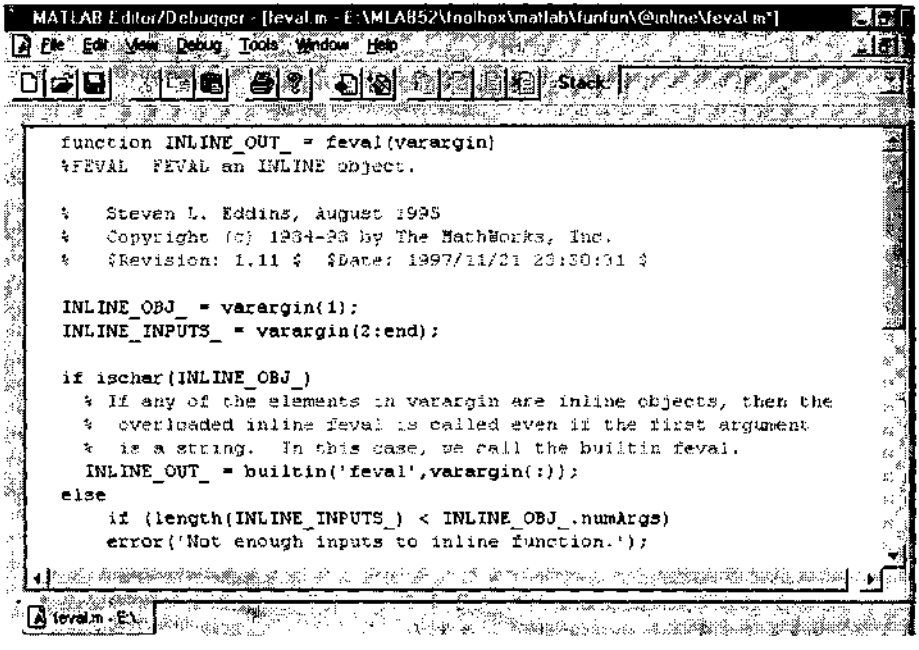


Рис. 2.2

Переопределяемые методы: help axischild/edit.m.

PCODE **Создать P-файл псевдокода**

Синтаксис:

pcode <*имя_М-файла*>
pcode <*имя_М-файла 1*> <*имя_М-файла 2*>...
pcode *.m
pcode... -inplace

Описание:

Команда pcode <*имя_М-файла*> выполняет грамматический анализ и преобразовывает М-файл с указанным именем в Р-файл псевдокода. Исходный М-файл должен находиться в пути доступа системы MATLAB. Р-файлы записываются в текущий рабочий каталог.

Команда pcode *<имя_М-файла 1> <имя_М-файла 2>...* преобразовывает все перечисленные М-файлы в Р-файлы и записывает их в текущем рабочем каталоге.

Команда pcode *.m преобразовывает все М-файлы в текущем рабочем каталоге в Р-файлы и записывает их в этот же каталог.

Команды pcode... -inplace с опцией inplace записывают Р-файлы в тот же каталог, где размещены преобразовываемые М-файлы.

Сопутствующие команды: HTML-справка.

MEX **Откомпилировать MEX-файл**

Синтаксис:

mex *<имя_внешней_программы> <опции>*
errorcode = mex(varargin)

Описание:

Хотя система MATLAB и обеспечивает полное и самодостаточное операционное окружение для программирования и обработки данных, в ряде случаев оказывается необходимым использовать внешние данные и внешние программы, написанные на языках C и Fortran. Вызов внешних программ, импорт и экспорт данных, установление связей клиент-сервер системы MATLAB с другими приложениями поддерживаются специальным интерфейсом прикладных программ API (Application Program Interface).

Для вызова программ на языках C или Fortran предложен механизм MEX-файлов, которые представляют собой программы-связки для обеспечения динамического вызова разноязыковых модулей. Эти программы создаются на основе программ, написанных на языках C или Fortran, и им присваивается расширение в зависимости от используемой платформы. Для платформы Windows таким расширением является dll.

Команда mex *<имя_внешней_программы> <опции>* создает MEX-файл с расширением dll для платформы Windows. Опции могут принимать следующие значения:

Опция	*Назначение*
-argcheck	Добавить код для проверки правильности входных аргументов (только для функций на языке C)
-c	Только компилировать
-D*<имя>*	Задать макроимя для препроцессора
-f *<путь_доступа>*	Использовать файл опций; если он не размещен в текущем каталоге, надо в параметре <file> указать полный путь
-g	Построить MEX-файл с включением символов отладки
-h[elp]	Напечатать заданное сообщение
-I*<путь_доступа>*	Включить параметр *<путь_доступа>* в опции компилятора
-O	Оптимизировать MEX-файл
-output *<имя>*	Создать MEX-файл с именем *<имя>*; соответствующее расширение формируется автоматически

-setup	Указать место размещения компилятора
-U<*имя*>	Отменить макроимя для препроцессора
-V4	Создать MEX-файл, совместимый с версией MATLAB 4
-v	Распечатать все параметры компилятора и компоновщика

Перечисленные опции можно оформить в виде специального файла опций mexopts.bat и указать на его использование единственной опцией -f. Для его написания используется язык сценариев Perl.

Функция errorcode = mex(varargin) создает MEX-файл с расширением dll и в случае появления ошибок накапливает их в переменной errorcode. Аргументы и опции командной формы этой функции должны быть представлены в виде переменной varargin.

Сопутствующие команды: HTML-справка.

QUIT, EXIT — **Завершение работы в системе MATLAB**

Синтаксис:

```
quit
exit
```

Описание:

Команда quit завершает работу системы MATLAB.

Команда exit служит для совместимости с более ранними версиями.

При завершении на экран выводится информация о том, какое число операций с плавающей запятой было выполнено за сеанс работы.

Сопутствующие команды: SAVE.

Необходимо помнить, что прекращение сеанса работы в системе MATLAB вызывает потерю текущих переменных, размещенных в рабочей области. Рабочую область перед выходом из системы можно сохранить для дальнейшего использования с помощью команды save.

Управление рабочей областью переменных

WHO, WHOS — **Вывод списков текущих переменных**

Синтаксис:

```
who                          whos
who global                   whos global
who -file <имя_MAT-файла>    whos -file <имя_MAT-файла>
who ... var1 var2            whos ... var1 var2
s = who(...)               s = whos(...)
```

Описание:

Команда who выводит список переменных текущей рабочей области.

Команда whos выводит подробную информацию относительно текущих переменных, включая имя, размеры и число элементов используемых массивов, длину в байтах, тип матрицы (плотная/редкая, комплексная/действительная).

Команды who global и whos global выводят списки глобальных переменных в глобальной рабочей области.

Команды who –file <*имя_МАТ-файла*> и whos –file <*имя_МАТ-файла*> выводят списки переменных в указанном МАТ-файле.

Команды who ... var1 var2 и whos ... var1 var2 выводят информацию только относительно указанных переменных. Групповой символ * можно использовать для того, чтобы вывести информацию о переменных, отвечающих некоторому шаблону, например: A*.

Рекомендуется использовать функциональные формы вида who('–file', <*имя_МАТ-файла*>, var1, var2), whos('–file', <*имя_МАТ-файла*>, var1, var2), если имена МАТ-файлов и переменных записаны в виде строк.

Функции вида s = who(...) возвращают массив ячеек, в котором записаны имена используемых переменных.

Функции вида s = whos(...) возвращают массив записей со следующими полями:

Поле	*Описание*
name	Имя переменной
bytes	Длина, байт
class	Класс переменной

Примеры:

Рассмотрим примеры использования командной и функциональной форм операторов who и whos:

```
who
Your variables are:
A     C     s
B     D     s1
```

```
s = who
s =
  'A'
  'B'
  'C'
  'D'
  's'
  's1'
```

```
whos
 Name   Size    Bytes    Class
 A      3x6     144      double array
 B      3x6     288      double array (complex)
 C      4x9     288      double array
 D      4x9     88       sparse array
 s      6x1     566      cell array
 s1     5x1     2154     struct array
Grand total is 157 elements using 3528 bytes
```

Общее количество элементов - 157, *использовано* 3528 *байт*

```
s = whos
s =
4x1 struct array with fields:
  name
  size
  bytes
  class
```

Сопутствующие команды: DIR, EXIST, HELP, WHAT.

SAVE **Выгрузка переменных из рабочей области на диск**

Синтаксис:

save
save <*имя_файла*>
save <*имя_файла*> <*переменные*>
save <*имя_файла*> <*опции*>
save <*имя_файла*> <*переменные*> <*опции*>

Описание:

Команда save выгружает все переменные рабочей области в двоичном формате в специальный файл с именем matlab.mat. Данные могут быть загружены в рабочую область командой load.

Команда save <*имя_файла*> выгружает все переменные из рабочей области в двоичный файл с именем <*имя_файла*>.mat. Если в качестве имени файла указано stdio, то команда save посылает данные на стандартное устройство вывода.

Команда save <*имя_файла*> <*переменные*> выгружает только массивы указанных переменных в двоичный файл с именем <*имя_файла*>.mat.

Команды save <*имя_файла*> <*опции*> и save <*имя_файла*> <*переменные*> <*опции*> записывают данные в ASCII-коде вместо двоичного, используя следующие опции:

Опция	*Формат данных*
–ascii	8-символьный ASCII-формат
–ascii –double	16-символьный ASCII-формат
–ascii –tabs	8-символьный ASCII-формат с табуляцией
–ascii –double –tabs	16-символьный ASCII-формат с табуляцией

Переменные, сохраненные в ASCII-формате, будут тождественны единственной переменной, которая совпадает с именем ASCII-файла. Следовательно, при загрузке такого файла в рабочей области появится только одна переменная и, чтобы обратиться к индивидуальным переменным, необходимо использовать оператор двоеточия.

При сохранении комплекснозначных данных в ASCII-формате мнимая часть данных будет утеряна, поскольку нельзя запомнить нечисловое значение 'i'.

Двоичные форматы, используемые в MAT-файлах, зависят от размера и типа массива. Массивы с нецелочисленными элементами и с количеством элементов менее 10000 сохраняются в формате с плавающей точкой 8 байт/элемент. Для целочисленных массивов с количеством элементов более 10000 могут использоваться компактные форматы с 1, 2 или 4 байтами на элемент, как это показано в таблице.

Значение элемента массива	*Количество байт на элемент*
от 0 до 255	1
от 0 до 65535	2
от -32767 до 32767	2
от $-2^{31}+1$ до $2^{31}-1$	4
для других значений	8

Библиотеки интерфейса прикладных программ API системы MATLAB содержат необходимые программы на языках C и Fortran для чтения и записи MAT-файлов из внешних программ.

Сопутствующие команды: FPRINTF, FWRITE, LOAD.

LOAD **Загрузка переменных с диска в рабочую область**

Синтаксис:

load
load <*имя_файла*>
load (<*имя_файла*>)
load <*имя_файла*>.***
load <*имя_файла*> –ascii
load <*имя_файла*> –mat
S = load(...)

Описание:

Команда load считывает данные из файла matlab.mat, если он создан командой save.

Команды load <*имя_файла*> и load(<*имя_файла*>) выполняют одинаковые операции, загружая переменные из MAT-файла <*имя_ файла*>.mat. Если вместо имени файла указать *stdio*, то команда load будет выполнять чтение со стандартного устройства ввода.

Команда load <*имя_файла*>.*** читает ASCII-файл с расширением .***, который должен представлять собой прямоугольный массив числовых данных, отделенных пробелами и размещаемых в m строках с n значениями в каждой строке. В результате образуется массив размера m×n с именем <*имя_файла*>. Чтобы обратиться к индивидуальным переменным, необходимо использовать оператор двоеточия. ASCII-файлы могут включать строки комментария, помеченные символом %.

Команды load <*имя_файла*> -ascii и load <*имя_файла*> -mat могут использоваться для принудительной загрузки файлов соответственно в форматах ASCII и MAT.

Функция S = load(...) возвращает содержимое MAT-файла в виде массива записей вместо загрузки переменных в рабочую область. Имена полей соответствуют именам переменных. В случае ASCII-файла S - это числовой массив.

Сопутствующие команды: FPRINTF, FSCANF, PARTIALPATH, SAVE, SPCONVERT.

Команды save и load накапливают и считывают данные с диска. Они могут также импортировать и экспортировать числовые массивы в виде ASCII-файлов.

MAT-файлы записываются в двоичном формате удвоенной точности с помощью команды save и считываются командой load. Они являются переносимыми, то есть могут быть созданы на одном компьютере, а прочитаны на другом, даже с иным форматом плавающей точки, сохраняя при этом точность и диапазон представления, насколько это допускает используемый формат. Эти данные могут использоваться программами и вне среды системы MATLAB.

MEMORY **Информация об ограничениях по памяти**

Синтаксис:

memory

Описание:

Команда memory возвращает следующую информацию о возможных ограничениях по памяти в системе MATLAB.

Если в сеансе работы появилось сообщение об ошибке Out of memory (Недостаточно памяти), то это означает, что отсутствует память для записи новых переменных. В этом случае, прежде чем продолжить выполнение, вам необходимо освободить память. Для этого можно использовать два способа: первый - удалить ряд переменных с помощью команды clear, второй - применить команду pack, чтобы удалить "мусор" и освободить непрерывные области памяти для записи переменных. Кроме того, средствами системы Windows можно увеличить размер виртуальной памяти.

Сопутствующие команды: CLEAR, PACK.

CLEAR **Освобождение рабочей области памяти**

Синтаксис:

clear
clear <*имя*>
clear <*имя 1*> <*имя 2*> <*имя 3*> ...
clear global <*имя*>
clear <*ключ*>

Описание:

Команда clear удаляет все переменные из рабочей области.

Команда clear <*имя*> удаляет переменную или функцию с именем X из рабочей области. Если X глобальная переменная, то clear X удаляет X из текущей рабочей области, но оставляет ее доступной для любой функции, где эта переменная объявлена глобальной. Допустимо использовать частичный путь доступа. Групповой символ * можно использовать для того, чтобы удалить все переменные, отвечающие некоторому шаблону, например: A*.

Команда clear <*имя 1*> <*имя 2*> <*имя 3*> ... удаляет из рабочей области только переменные и функции с указанными именами.

Команда clear global <*имя*> удаляет глобальную переменную с указанным именем.

Команда clear <*ключ*> удаляет те или иные переменные и функции в зависимости от значения ключа:

Значение ключа	*Действие ключа*
functions	Удаляет из рабочей области все М-файлы
variables	Удаляет из рабочей области все переменные
mex	Удаляет из рабочей области все MEX-файлы
global	Удаляет из рабочей области все глобальные переменные
all	Удаляет из рабочей области все типы переменных, функций, все MEX-файлы, оставляя ее пустой

Допустимо использовать функциональную форму вида clear(...).

Сопутствующие команды: MLOCK, MUNLOCK, PACK.

PACK **Дефрагментация рабочей области памяти**

Синтаксис:

pack

pack <*имя_файла*>

Описание:

Команда pack освобождает память, размещая переменные в минимально возможном объеме.

Команда pack <*имя_файла*> использует файл с указанным именем в качестве временного файла для записи переменных; если такой файл не указан, используется временный файл pack.tmp.

Команда pack не влияет на размер памяти, выделенный процессу исполнения программы MATLAB; чтобы освободить эту память, необходимо выйти из системы. Поскольку система MATLAB использует метод управления памятью, называемый кучей, то в течение длительного сеанса работы память становится фрагментированной, то есть появляется много фрагментов свободного пространства, но отсутствуют непрерывные участки для сохранения больших по размеру массивов.

Если получено сообщение Out of memory (Недостаточно памяти), то команда pack может освободить участок памяти, не удаляя при этом переменных.

Команда pack освобождает память, выполняя так называемую сборку "мусора"; она сохраняет все переменные на диске во временном файле pack.tmp, очищает память и затем вновь загружает переменные на диск из файла pack.tmp, на последней фазе удаляется файл pack.tmp.

Если применение команды pack не дает результатов, следует попробовать удалить ряд переменных; если и этого недостаточно, то следует увеличить размер памяти, используемой для подкачки (свопинга). Для этого в панели управления (Control Panel) откройте пиктограмму 386 Enhanced, нажмите кнопку Virtual Memory и увеличьте размер памяти.

Сопутствующие команды: CLEAR.

WORKSPACE **Запуск Workspace Browser**

Синтаксис:

workspace

Описание:

Команда workspace запускает средство просмотра рабочей области Workspace Browser.

Сопутствующие команды: EDIT, PATHTOOL.

Управление путями доступа

PATHDEF **Путь доступа по умолчанию**

Синтаксис:

p = pathdef

Описание:

Функция p = pathdef присваивает строковой переменной p список путей доступа, используемый по умолчанию в системе MATLAB. Этот список соответствует встроенной функции matlabpath, которая используется в главном файле запуска matlabrc. Это специальный файл, написанный на фирме MathWorks, и он не должен изменяться.

Пример:

p = pathdef;

Результатом является строка, имеющая очень большую длину и не пригодная для вывода на экран. Для вывода списка путей доступа в наглядном для восприятия виде следует использовать команду matlabpath.

Сопутствующие команды: PATH, PATHSEP, MATLABPATH.

PATH **Управление списком путей доступа**

Синтаксис:

path
p = path

path(<*новый_путь_доступа*>)
path(path, <*новый_путь_доступа*>)
path(<*новый_путь_доступа*>, path)

Описание:

Команда path выводит на экран список путей доступа в системе MATLAB. Этот список соответствует переменной MATLABPATH, которая устанавливается командой matlabrc или индивидуальной программой запуска startup.

Команда p = path возвращает строку, содержащую список путей доступа.

Команда path(<*новый_путь_доступа*>) заменяет существующий список списком <*новый_путь_доступа*>.

Команда path(path, <*новый_путь_доступа*>) добавляет новый путь доступа в конец списка путей доступа path.

Команда path(<*новый_путь_доступа*>, path) добавляет новый путь доступа в начало списка путей доступа path.

Сопутствующие команды: ADDPATH, CD, DIR, RMPATH, WHAT.

PATHSEP **Разделитель, используемый в списке путей доступа**

Синтаксис:

p = pathsep

Описание:

Функция p = pathsep присваивает строковой переменной p тип разделителя, используемый на данной платформе. Разделитель - это символ, который употребляется для отделения друг от друга каталогов в списке путей доступа.

Пример:

p = pathsep
p = ;

На платформе PC разделителем является точка с запятой.

Сопутствующие команды: FILESEP, FULLFILE, PATH.

PARTIALPATH **Частичный путь доступа**

Описание:

Функция p = pathsep присваивает строковой переменной p тип разделителя, используемый на данной платформе. Разделитель - это символ, который применяется для отделения друг от друга каталогов в списке путей доступа.

Частичный путь доступа - это часть полного списка путей доступа MATLABPATH, которая, как правило, связана с частными каталогами или каталогами методов, которые обычно скрыты или поиск в них ограничен из-за того, что в этих каталогах могут присутствовать файлы с дублирующими именами.

Частичный путь доступа включает последнюю или несколько последних компонентов полного пути доступа, например: matfun/trace, private/children, inline/formula, demos/clown.mat. Использование символа @ в каталоге метода необязательно, и поэтому частичный путь вида funfun/inline/formula является правильным.

Многие команды системы MATLAB допускают использование частичных путей доступа, в том числе help, type, load, exist, what, which, edit, profile, dbtype, dbstop, dbclear, fopen.

Частичные пути позволяют упростить поиск пакетов прикладных программ или компонентов системы MATLAB при переносе системы с одного компьютера на другой, поскольку частичные пути независимы от места установки системы.

Сопутствующие команды: MATLABPATH, PATH.

GENPATH **Добавить к списку путей доступа каталог**

Синтаксис:

p = genpath
p = genpath(dir)

Описание:

Функция p = genpath формирует новый список путей доступа, добавляя к прежнему списку пути, которые принадлежат каталогу MATLABROOT/toolbox.

Функция p = genpath(<*каталог*>) формирует новый список путей доступа, добавляя к прежнему списку пути, которые принадлежат каталогу <*каталог*>.

Сопутствующие команды: FILESEP, FULLFILE, PATH.

PATH2RC **Сохранить список путей доступа**

Синтаксис:

path2rc
path2rc <*выходной_файл*>

Описание:

Функция path2rc сохраняет текущий список путей доступа в виде файла pathdef.m, который будет исполнен при запуске системы.

Функция path2rc <*выходной_файл*> сохраняет текущий список путей доступа в специальном выходном файле.

Указанные функции возвращают следующий признак:

Признак	*Значение*
0	Файл сохранен успешно
1	Файл не может быть сохранен
2	Файл pathdef.m не найден
3	Файл pathdef.m найден, но не может быть прочитан

Сопутствующие команды: FILESEP, FULLFILE, PATH.

MATLABROOT **Корневой каталог системы MATLAB**

Синтаксис:

s = matlabroot

Описание:

Функция s = matlabroot возвращает строку с именем каталога, в котором установлена система MATLAB.

Пример:

s = matlabroot
s = E:\MLAB52

Сопутствующие команды: FULLFILE.

ADDPATH **Добавить каталог к списку путей доступа**

Синтаксис:

addpath <*каталог*>
addpath <*каталог1*> <*каталог2*> <*каталог3*> ...
addpath ... –end | -begin

Описание:

Команда addpath <*каталог*> добавляет указанный каталог к началу списка путей доступа. Если имя каталога содержит пробелы, следует использовать функциональную форму оператора addpath('<*каталог*>').

Команда addpath <*каталог1*> <*каталог2*> <*каталог3*> ... добавляет указанные каталоги к началу списка путей доступа.

Команда addpath ... -begin | –end в зависимости от значения флага добавляет указанные каталоги в начало (-begin, либо 0) или в конец (-end, либо 1) списка путей доступа.

Сопутствующие команды: PATH, RMPATH.

RMPATH **Удалить каталог из списка путей доступа**

Синтаксис:

rmpath <*каталог*>
rmpath <*каталог1*> <*каталог2*> <*каталог3*> ...

Описание:

Команда rmpath <*каталог*> удаляет указанный каталог из списка путей доступа. Если имя каталога содержит пробелы, следует использовать функциональную форму оператора rmpath('<*каталог*>').

Команда rmpath <*каталог1*> <*каталог2*> <*каталог3*> ... удаляет указанные каталоги из списка путей доступа.

Рекомендуется использовать функциональную форму оператора rmpath('<*каталог1*>', '<*каталог2*>', ...), если каталоги записаны в виде строковых выражений.

Сопутствующие команды: ADDPATH, PATH.

FINDDEMO **Определить расположение демонстрационных файлов**

Синтаксис:

finddemo

Описание:

Команда finddemo отыскивает пути доступа, содержащие файлы Demos.m и Demos.mat, а также выводит список ППП и комплектов средств, содержащих файлы Demo.m.

Сопутствующие команды: HTML-справка.

EDITPATH, PATHTOOL **Запуск средства редактирования и просмотра путей доступа Path Browser**

Синтаксис:

editpath
pathtool

Описание:

Команды editpath и pathtool в среде Windows выполняют одинаковые функции, осуществляя запуск средства редактирования и просмотра путей доступа Path Browser.

Сопутствующие команды: EDIT, WORKSPACE.

Управление командным окном

CLC **Очистка командного окна**

Синтаксис:

clc

Описание:

Команда clc очищает командное окно и возвращает курсор в левый верхний угол экрана.

Сопутствующие команды: HOME.

HOME **Возвращение курсора**

Синтаксис:

home

Описание:

Команда home возвращает курсор в верхний левый угол экрана.

Сопутствующие команды: CLC.

ECHO **Вывод на экран текста Script-файла**

Синтаксис:

echo on | off | *<без параметров>*
echo *<имя файла>* on | off | *<без параметров>*
echo on | off all

Описание:

Команда echo on включает режим вывода на экран текста Script-файла.

Команда echo off отменяет предыдущую команду.

Команда echo управляет режимом echo при выполнении файла-сценария.

Команда echo *<имя_файла>*on выводит на экран текст М-файла с указанным именем.

Команда echo *<имя_файла>*off отменяет предыдущую команду.

Команда echo *<имя_файла>* переключает режим на противоположный.

Команда echo on all включает режим вывода на экран содержимого всех файлов-функций.

Команда echo off all отменяет режим echo on all.

Определить текущее состояние режима echo можно с помощью команды get(0, 'echo').

Сопутствующие понятия: FUNCTION, SCRIPT.

MORE **Управление выводом информации на экран**

Синтаксис:

more on | off
more(n)

Описание:

Команда more on включает режим постраничного вывода.

Команда more off отключает режим постраничного вывода.

Команда more(n) определяет размер страницы длиной n строк.

По умолчанию приняты значения off и n = 23.

Когда режим more включен, реализуется постраничный вывод и переход к следующей строке осуществляется нажатием клавиши Return; для просмотра следующей страницы надо нажать клавишу пробела. Для выхода из режима просмотра используйте клавишу q.

Форматы вывода числовой информации

Формат представления чисел на экране дисплея по желанию пользователя можно регулировать с помощью команды format. Эта команда определяет лишь форму вывода результатов и не влияет на точность вычислений, а также на формат представления и хранения чисел. (В системе MATLAB вычисления всегда выполняются с двойной точностью.)

Если все элементы массива являются целыми числами, то они отображаются на экране в формате без десятичной точки (вне зависимости от текущего формата вывода). Если же хотя бы один элемент не является целым числом, то все элементы такого массива будут выведены на экран в текущем выходном формате. В системе MATLAB возможно задание нескольких выходных форматов.

По умолчанию при загрузке системы устанавливается формат short. Он отображает только 5 значащих десятичных цифр числа. Другие форматы позволяют отобразить большее число десятичных цифр или используют экспоненциальное представление чисел.

FORMAT **Форматы вывода**

Синтаксис:

format	format short g	format +
format short	format long g	format compact
format long	format hex	format loose
format short e	format bank	
format long e	format rat	

Описание:

Команда	*Формат*	*Пример*
format	По умолчанию, соответствует format short	3.1416
format short	Короткое число с фиксированной точкой: 5 десятичных цифр	3.1416
format long	Длинное число с фиксированной точкой: 15 десятичных цифр	3.14159265358979
format short e	Короткое число с плавающей точкой: 5 десятичных цифр	3.1416e+00
format long e	Длинное число с плавающей точкой: 15 десятичных цифр	3.141592653589793e+000
format short g	Лучшая форма из 5 десятичных цифр	3.1416
format long g	Лучшая форма из 15 десятичных цифр	3.14159265358979
format hex	Шестнадцатеричное число	400921fb54442d18
format bank	Коммерческий формат	3.14
format rat	Рациональное число	355/113

format +	Символическое отображение числа: положительное + отрицательное - нуль **пробел**	+
format compact	Подавление пробела между строками	
format loose	Восстановление пробела между строками	

Сопутствующие команды: FPRINTF, NUM2STR, RAT, SPRINTF, SPY.

DIARY **Ведение дневника**

Синтаксис:

diary <*имя файла*>

diary on | off | <*без параметров*>

Описание:

Команда diary <*имя файла*> включает режим записи в файл с заданным именем последовательности выполненных пользователем команд, а также результатов их выполнения, кроме графических изображений.

Команда diary on включает режим записи.

Команда diary off выключает режим записи.

Команда diary <*без параметров*> переключает состояние дневника (с on на off и наоборот).

Эта команда позволяет помещать в дневник отдельные фрагменты выполняемых действий.

Сопутствующие команды: HTML-справка.

Работа с файлами и операционной системой

PWD **Текущий каталог системы MATLAB**

Синтаксис:

pwd

s = pwd

Описание:

Команда pwd выводит на терминал текущий каталог при работе в системе MATLAB.

Функция s = pwd возвращает текущий каталог в виде строковой переменной.

Сопутствующие команды: CD.

CD **Состояние и изменение текущего каталога**

Синтаксис:

cd
cd ..
cd <*путь_доступа*>

Описание:

Команда cd выводит на экран пути доступа к текущему каталогу.

Команда cd.. выводит на экран каталог с иерархией на единицу выше.

Команда cd <*путь_доступа*> изменяет текущий каталог на каталог, определяемый пользователем.

Сопутствующие команды: PWD.

DIR **Вывод на экран листинга каталога**

Синтаксис:

dir
dir<*имя_каталога*>
D = dir<*имя_каталога*>

Описание:

Команда dir выводит листинг текущего каталога.

Команда dir <*имя_каталога*> выводит на экран список файлов, содержащихся в каталоге. Можно указывать путь доступа к файлу и шаблон для имен файлов текущего каталога.

Функция D = dir('<*имя_каталога*>') возвращает массив записей со следующими полями:

Поле	*Назначение*
name	Имя файла
date	Дата модификации
bytes	Размер в байтах
isdir	1 - если это каталог; 0 - в остальных случаях

Пример:

```
d=dir('m*.m*')
d =
2x1 struct array with fields:
   name
   date
   bytes
   isdir
```

```
d.name                d.date
ans = mn.mat          ans = 07-Jul-1998 17:29:08
ans = myf.my          ans = 10-Jul-1998 08:27:46
d.bytes               d.isdir
ans =   328           ans =    0
ans = 1642            ans =    0
```

Сопутствующие команды: !, CD, DELETE, TYPE, WHAT.

COPYFILE **Скопировать файл**

Синтаксис:

copyfile(<*файл_источник*>, <*новый_файл*>)
copyfile(<*файл_источник*>, <*новый_файл*>, 'writable')
status = copyfile(...)
[status, msg] = copyfile(...)

Описание:

Команда copyfile(<*файл_источник*>, <*новый_файл*>) копирует файл-источник в новый файл. Аргументы <*файл_источник*> и <*новый_файл*> могут быть заданы в виде полных или частичных путей доступа.

Команда copyfile(<*файл_источник*>, <*новый_файл*>, 'writable') подтверждает, что новый файл доступен для записи.

Функция status = copyfile(...) возвращает 1, если копирование успешно, и 0 - в остальных случаях.

Функция [status, msg] = copyfile(...) возвращает, кроме того, сообщение об ошибке, если последняя имела место.

Сопутствующие команды: DELETE, MKDIR.

DELETE **Удаление файлов и графических объектов**

Синтаксис:

delete <*имя файла*>
delete (h)

Описание:

Команда delete <*имя файла*> удаляет заданный файл с диска.

Команда delete(h) удаляет графический объект с дескриптором h. Если объект - окно, то оно закрывается и удаляется без подтверждения.

Переопределяемые методы:

help char/delete.m
help scribehgobj/delete.m
help scribehandle/delete.m
help hgbin/delete.m
help editrect/delete.m
help arrowline/delete.m
help activex/delete.m

GETENV **Информация о переменных среды DOS**

Синтаксис:

s = getenv <*переменная среды*>

Описание:

Функция s = getenv <*переменная среды*> возвращает текущую информацию о переменной среды DOS и результат в виде строки s.

Пример:

Функция s = getenv('temp') возвращает значение s = C:\DOS.
Функция s = getenv('prompt') возвращает значение s = pg.

Сопутствующие команды: HTML-справка.

MKDIR **Создать каталог**

Синтаксис:

mkdir(<*имя_каталога*>)
mkdir(<*родительский_каталог*>, <*новый_каталог*>)
status = mkdir(...)
[status,msg] = mkdir(...)

Описание:

Команда mkdir(<*имя_каталога*>) создает каталог с указанным именем в текущем каталоге.

Команда mkdir(<*родительский_каталог*>, <*новый_каталог*>) создает новый каталог в уже существующем родительском каталоге.

Функция status = mkdir(...) возвращает 1, если каталог успешно создан, 2 - если такой каталог уже существует, и 0 - в остальных случаях.

Функция [status, msg] = mkdir(...) возвращает, кроме того, сообщение об ошибке, если последняя имела место.

Сопутствующие команды: COPYFILE.

DOS, ! **Выполнить команду DOS и возвратить результат**

Синтаксис:

dos <*команда_DOS*>
! <*команда_DOS*>
[status, result] = dos('<*команда*>', '-echo')
[status, result] = dos('<*команда*>' | '<*команда*> &')

Описание:

Команды dos <*команда_DOS*> и ! <*команда_DOS*> выполняют команды DOS.

Функция [status, result] = dos('<*команда*>', '-echo') в среде Windows вызывает оболочку DOS, чтобы выполнить заданную команду. Могут выполняться как команды DOS, так и команды Windows, однако способы представления результатов будут отличаться. Команды DOS всегда возвращают результат в виде переменной. Они всегда выполняются активизируя пиктограмму DOS или командное окно, за исключением некоторых случаев, перечисленных ниже.

Команды DOS никогда не выполняются в фоновом режиме, и, кроме того, MATLAB будет ожидать завершения операций вывода, прежде чем продолжить вычисления.

В свою очередь, команды Windows могут исполняться в фоновом режиме.

Команды DOS допускают использование второго параметра '-echo'. Этот параметр заставляет вывести результаты в командное окно, даже если предписывался вывод только в виде переменных.

Использование символа & имеет следующее назначение. При выполнении команд DOS он вызывает открытие окна DOS, в отсутствие этого символа активизируется только пиктограмма DOS. При выполнении команд Windows он вызывает исполнение программы в фоновом режиме.

Пример:

[s, w] = dos('dir')

Эта команда выводит на экран листинг каталога, возвращает переменную s, равную нулю, а в строковой переменной w сохраняет листинг.

dos('edit &')

Эта команда открывает редактор DOS в окне DOS.

dos('notepad file.m &')

Эта команда открывает редактор notepad и возвращает управление системе MATLAB.

[s, w] = dos('foo')

Эта команда возвращает значение s, равное нулю, поскольку оболочка DOS вызвана правильно, однако в строковой переменной w будет содержаться сообщение об ошибке, поскольку "foo" не является командой DOS.

[s, w] = dos('dir', '-echo');

Эта команда будет выводить результаты в командное окно, поскольку опция -echo подавляет действие символа ';'.

Сопутствующие команды: HTML-справка.

COMPUTER **Получение информации о компьютере**

Синтаксис:

computer

[c, maxsize] = computer

Описание:

Команда computer возвращает строку с информацией о типе компьютера, на котором установлена система MATLAB.

Функция [c, maxsize] = computer возвращает следующую информацию:

- c - строка с информацией о типе компьютера;
- maxsize - целое число, указывающее на максимально допустимое число элементов матрицы для данной версии MATLAB. (Дополнительные ограничения определяются возможностями операционной системы.)

Формируются следующие сообщения о типе компьютера:

Сообщение	*Тип компьютера*
PCWIN	MS-Windows
MAC2	Macintosh
SUN4	Sun SPARC
SOL2	Solaris 2
HP700	HP 9000/700
SGI	Silicon Graphics
SGI64	Silicon Graphics R8000
IBM_RS	IBM RS6000
ALPHA	Dec Alpha
AXP_VMSG	Alpha VMS G_float
AXP_VMSIEEE	Alpha VMS IEEE
LNX86	Linux Intel
VAX_VMSG	VAX/VMS G_float
VAX_VMSD	VAX/VMS D_float

Пример:

После выполнения команды

[c, maxsize] = computer

в рабочем окне MATLAB появятся сообщения

c = PCWIN

maxsize = 268435455

Сопутствующие команды: ISIEEE, ISPPC, ISUNIX, ISVMS.

TEMPDIR **Имя рабочего каталога DOS**

Синтаксис:

tempdir

Описание:

Команда tempdir возвращает имя рабочего каталога DOS, если он существует.

Сопутствующие команды: TEMPNAME, FULLFILE.

TEMPNAME **Имя временного файла**

Синтаксис:

tempname

Описание:

Команда tempname возвращает уникальное имя, которое может быть использовано в качестве имени временного файла.

Сопутствующие команды: TEMPDIR.

FULLFILE **Сформировать полное имя файла из частей**

Синтаксис:

fullfile(d1, d2, ... , <*имя_файла*>)

Описание:

Команда fullfile(d1, d2, ... , <*имя_файла*>) формирует путь доступа к файлу, используя имена каталогов d1, d2, ... и имя файла. Это в основном соответствует следующей конструкции:

F = [d1 <*разделитель*> d2 <*разделитель*> ... <*разделитель*> <*имя_файла*>].

Пример:

Сформировать независимый от используемого компьютера путь доступа к файлу Contents.m:

fullfile(matlabroot,'toolbox','matlab','general','Contents.m')

ans = E:\MLAB52\toolbox\matlab\general\Contents.m

Сформировать независимый от используемого компьютера путь доступа к каталогу matlab:

fullfile(matlabroot,'toolbox','matlab','')

ans = E:\MLAB52\toolbox\matlab

Сопутствующие команды: FILEPARTS, FILESEP, PATHSEP.

FILESEP **Разделитель каталогов для данного компьютера**

Синтаксис:

f = filesep

Описание:

Функция f = filesep возвращает разделитель каталогов, используемый в данном операционном окружении.

Сопутствующие команды: FULLFILE, PATHSEP.

FILEPARTS **Выделить составляющие пути доступа**

Синтаксис:

[path, name, ext] = fileparts(<*путь_доступа*>)

Описание:

Функция [path, name, ext] = fileparts(<*путь_доступа*>) возвращает путь доступа, имя файла и его расширение.

Пример:

```
[path,name,ext] = fileparts(' E:\MLAB52\toolbox\matlab\general\Contents.m')
path = E:\MLAB52\toolbox\matlab\general
name = Contents
ext = .m
```

Сопутствующие команды: FILESEP, FULLFILE, PATHSEP.

3. ТИПЫ ДАННЫХ И ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД

В системе MATLAB определено 6 базовых типов данных, каждый из которых является тем или иным видом массива. Шесть классов - это double, sparse, uint8, char, cell и struct. Принадлежность того или иного объекта системы MATLAB к одному из классов может быть представлена следующей схемой (рис. 3.1).

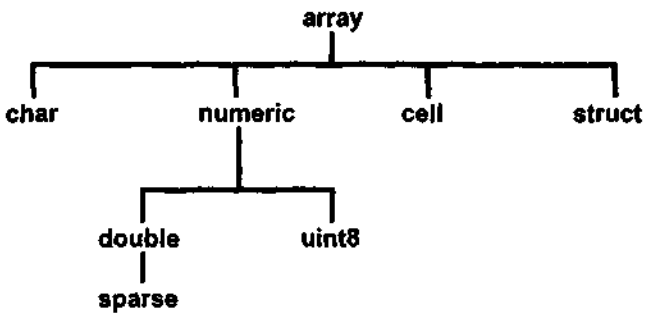


Рис. 3.1

Соединительные линии на схеме определяют принадлежность того или иного типа данных к одному или нескольким классам. Обратите внимание, что тип array - массив, находящийся в вершине схемы. Это означает, что все данные системы MATLAB являются массивами.

Чаще всего приходится иметь дело с двумя из этих классов: массивом чисел удвоенной точности (double) и массивом символов (char), или просто строкой. Это обусловлено тем, что вычисления в системе MATLAB выполняются с удвоенной точностью, и поэтому большинство функций работают с массивами чисел удвоенной точности, а также со строками.

Другие классы предназначены для таких специальных приложений, как работа с разреженными матрицами (sparse), обработка изображений (uint8), работа с массивами большой размерности (cell и struct).

Нельзя задать тип переменной numeric или array. Эти типы называются *виртуальными* и служат только для того, чтобы сгруппировать переменные, которые имеют общие атрибуты.

Тип uint8 предназначен для эффективного хранения данных в памяти. К данным этого типа можно применять только базовые операции индексации и изменения размеров, но нельзя выполнить никакой математической операции. Для этого такие массивы необходимо преобразовать в тип double.

Перечисленные на схеме типы данных называются *встроенными*. Для встроенных типов можно переопределять метод обработки, как это делается для объектов системы MATLAB. Например, чтобы задать операцию сортировки для массива типа uint8, необходимо создать метод (sort.m или sort.mex) и поместить его в специальный каталог @uint8.

Язык MATLAB позволяет создавать собственные типы данных UserObject и работать с ними по аналогии со встроенными типами.

Следующая таблица описывает типы данных более подробно.

Класс	*Пример*	*Описание*
Double	[1 2; 3 4] 5 + 6i	**Числовой массив** удвоенной точности (это наиболее распространенный тип переменной в системе MATLAB)
Char	'Привет'	**Массив символов** (каждый символ - длиной 16 бит), часто именуется *строкой*
Sparse	Speye(5)	**Разреженная матрица** удвоенной точности (только двумерная). Разреженная структура применяется для хранения матриц с небольшим количеством ненулевых элементов, что позволяет использовать лишь небольшую часть памяти, необходимой для хранения полной матрицы. Разреженные матрицы требуют применения специальных методов для решения задач
Cell	{ 17 'привет' eye (2)}	**Массив ячеек.** Элементы этого массива содержат другие массивы. Массивы ячеек позволяют объединить связанные данные, возможно различных размеров, в единую структуру
Struct	A.day = 12; A.color = 'Red'; A.mat = magic(3);	**Массив записей.** Он включает имена полей. Поля сами могут содержать массивы. Подобно массивам ячеек, массивы записей объединяют связанные данные и информацию о них
Uint8	Uint8 (magic (3))	**Массив 8-разрядных целых чисел без знаков.** Он позволяет хранить целые числа в диапазоне от 0 до 255 в 1/8 части памяти, требуемой для массива удвоенной точности. Никакие математические операции для этих массивов не определены
UserObject	inline('sin(x)')	**Тип данных, определяемый пользователем**

Каждому типу данных соответствуют свои функции и операторы обработки, или, другими словами, *методы*. Дочерние типы данных, расположенные на диаграмме ниже родительского типа, поддержаны также и методами родителя. Следовательно, массив типа double поддержан методами, применяемыми для типа numeric.

В следующей таблице приведены некоторые из таких методов.

Класс	*Метод*
Массив array	Вычисление размера (size), длины (length), размерности (ndims), объединение массивов ([a b]), транспонирование (transpose), многомерная индексация (subsindex), переопределение (reshape) и перестановка (permute) размерностей многомерного массива
Массив ячеек cell	Индексация с использованием фигурных скобок $\{e_1,\ldots,e_n\}$ и разделением элементов списка запятыми
Строка Char	Строковые функции (strcmp, lower), автоматическое преобразование в тип double для применения методов класса double
Double	Арифметические и логические операции, математические функции, функции от матриц
Numeric	Поиск (find), обработка комплексных чисел (real, imag), формирование векторов, выделение строк, столбцов, подблоков массива, расширение скаляра
Sparse	Операции над разреженными матрицами
Массив записей Struct	Доступ к содержимому поля .field (разделитель элементов списка - запятая)
Uint8	Операция хранения (чаще всего используется с ППП Image Processing Toolbox)
UserObject	Определяется пользователем

Числовые и логические массивы

Массивы являются основными объектами системы MATLAB: в ранних версиях допускались только одномерные и двумерные массивы; в системе MATLAB 5 возможно использование многомерных массивов.

Основные характеристики

LOGICAL — **Преобразовать числовой массив в логический**

Синтаксис:

L = logical(A)

Описание:

Функция L = logical(A) возвращает массив, который может быть использован в качестве шаблона для логических тестов. Оператор A(L) равносилен конструкции A(FIND(L)) и возвращает значения элементов массива A только для тех индексов, для которых элементы L равны 1.

Логические массивы могут быть также созданы с помощью операций отношения ==, <, >, ~, ... и функций типа any, all, isnan, isinf, isfinite.

Примеры:

Пусть задан числовой массив

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

и логический массив

```
L = logical(eye(3))
L =
   1   0   0
   0   1   0
   0   0   1,
```

соразмерный с A.

Применим оператор

```
A(L)
ans =
1
5
9,
```

результатом которого являются выделенные из числового массива A диагональные элементы.

Попытка применить для индексирования числовой массив eye(3), численно совпадающий с L, приведет к ошибке

```
A(eye(3))
??? Index into matrix is negative or zero.
```

Матричный индекс отрицателен или равен нулю.

Сопутствующие функции: ISLOGICAL, <, <=, >, >=, ==, ~=.

ISNUMERIC **Проверить, является ли массив числовым**

Синтаксис:

```
k = isnumeric(A)
```

Описание:

Функция k = isnumeric(A) возвращает 1, если A числовой массив, и 0 - в противном случае. Массивы типа sparse, double, logical являются числовыми массивами numeric, в то время как массивы строк char, записей struct и ячеек cell таковыми не являются.

Сопутствующие функции: ISCELL, ISLOGICAL, ISOBJECT, ISSPARSE, ISSTRUCT.

ISLOGICAL **Проверить, является ли массив логическим**

Синтаксис:

k = islogical(A)

Описание:

Функция k = islogical(A) возвращает 1, если A логический массив, и 0 - в противном случае. Логический массив является числовым, но числовой массив не обязательно логический.

Сопутствующие функции: LOGICAL.

ISEMPTY **Проверить, является ли массив пустым**

Синтаксис:

k = isempty(A)

Описание:

Функция k = isempty(A) возвращает 1, если массив A пустой, и 0 - в противном случае. Массив A считается пустым, если выполняется условие prod(size(X))==0.

Начиная с версии 5.0 операторы отношения вида A == [ ] должны быть заменены функцией isempty(A).

Сопутствующие функции: LOGICAL.

ISEQUAL **Проверить, являются ли массивы равными**

Синтаксис:

k = isequal(A, B, ...)

Описание:

Функция k = isequal(A, B, ...) возвращает 1, если все входные массивы имеют одинаковое содержимое, то есть равны между собой, и 0 - в противном случае.

Сопутствующие функции: EQ.

Одномерные и двумерные массивы

: **Получить доступ к подблокам одномерных и двумерных массивов**

Синтаксис:

j : k	A(i1 : i2, j1 : j2)
j : i : k	A(n1 : n2)

Описание:

Оператор : очень часто используется при работе с системой MATLAB. Он применяется для формирования векторов и массивов или для выделения из них подвекторов и подблоков массива.

Формирование векторов:

j : k

если j >= k, это вектор вида [j j+1 j+2 ... k];

если j < k, это *пустой* вектор;

j : i : k

если j >= k, это вектор вида [j j+i j+2i... k];

если i < 0 и j < k или i > 0 и j > k, это *пустой* вектор.

Выделение подблоков:

A(i1 : i2, j1 : j2) - выделение подблока массива A со строками i1 : i2 и столбцами j1 : j2.

A(i, :) - *i-я строка массива* A;

A(:, j) - *j-й столбец массива* A.

Поскольку в языке MATLAB элементы массива упорядочены по столбцам, то допустимы операторы вида A(n1:n2), которые выделяют пронумерованные элементы с номера n1 до номера n2. Оператор A(:) записывает все элементы массива A в виде столбца.

Примеры:

D = 1:4

D = 1 2 3 4

E = 0 : .1 : .5

E = 0 0.1000 0.2000 0.3000 0.4000 0.5000

Сопутствующие функции: LINSPACE, LOGSPACE, RESHAPE.

[,], [;] **Объединение массивов**

Синтаксис:

[A, B, ...]

[A; B; ...]

Описание:

Оператор [A, B, ...] выполняет горизонтальное объединение конечного количества массивов, у которых количество строк должно быть одинаково. Оператор [A, B, ...] равносилен оператору [A B ...].

Оператор [A; B; ...] выполняет вертикальное объединение конечного количества массивов, у которых должно быть одинаковым количество столбцов.

Горизонтальное и вертикальное объединение может быть скомбинировано в одном операторе.

Оператор [A B; C] является допустимым, если количество строк массивов A и B одинаково, а количество столбцов массива C совпадает с суммой столбцов массивов A и B.

Оператор [A B; [C D]] является допустимым, если сумма столбцов массивов C и D совпадает с суммой столбцов массивов A и B.

Аналогичными приемами могут быть построены очень сложные конструкции.

Сопутствующие функции: HORZCAT, RESHAPE, VERTCAT.

RESHAPE **Преобразование размеров двумерного массива**

Синтаксис:

B = reshape(A, m, n)

Описание:

Функция B = reshape(A, m, n) возвращает двумерный массив размера m×n, сформированный из элементов массива A путем их последовательной выборки по столбцам. Если количество элементов массива A не равно произведению m * n, выводится сообщение об ошибке.

Алгоритм:

С помощью оператора : можно получить те же результаты, которые можно получить и с помощью функции reshape. Функция reshape выражается через оператор : следующим образом:

B = zeros(m, n); B(:) = A;

Пример:

Пусть $A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$.

Тогда reshape(A, 6, 2) = $\begin{bmatrix} 1 & 3 \\ 1 & 3 \\ 1 & 3 \\ 2 & 4 \\ 2 & 4 \\ 2 & 4 \end{bmatrix}$, а reshape(A, 2, 6) = $\begin{bmatrix} 1 & 1 & 2 & 3 & 3 & 4 \\ 1 & 2 & 2 & 3 & 4 & 4 \end{bmatrix}$.

Сопутствующие функции и операторы: FLIPLR, FLIPUD, ROT90, :.

REPMAT **Формирование двумерного массива из блоков**

Синтаксис:

B = repmat(A, m, n)

Описание:

Функция B = repmat(A, m, n) возвращает двумерный массив, сформированный из m × n блоков A, из них m блоков по вертикали и n блоков по горизонтали.

Обращение B = repmat(A, m, n) равносильно B = repmat(A, [m n]).

Пример:

```
repmat(magic(2), 2, 3)
ans =
   1   3   1   3   1   3
   4   2   4   2   4   2
   1   3   1   3   1   3
   4   2   4   2   4   2
repmat(NaN, 2, 3)
```

```
ans =
  NaN  NaN  NaN
  NaN  NaN  NaN
```

Сопутствующие функции и операторы: MESHGRID.

FIND **Поиск индексов и значений элементов массива по заданному условию**

Синтаксис:

k = find(x)	k = find(x<*условие*>)
[i, j] = find(X)	[i, j] = find(X<*условие*>)
[i, j, s] = find(X)	[i, j, s] = find(X<*условие*>)

Описание:

Функция k = find(x) возвращает индексы ненулевых элементов одномерного массива x.

Функция k = find(x<*условие*>) возвращает индексы элементов одномерного массива x, удовлетворяющие заданному условию.

Функция [i, j] = find(X) возвращает индексы ненулевых элементов двумерного массива X.

Функция [i, j] = find(X<*условие*>) возвращает индексы элементов двумерного массива X, удовлетворяющие заданному условию.

Функция [i, j, s] = find(X) возвращает кроме индексов ненулевых элементов двумерного массива X также и их значения в виде вектора s.

Функция [i, j, s] = find(X<*условие*>) возвращает кроме индексов элементов двумерного массива X, удовлетворяющих заданному условию, также и булев вектор s, состоящий из единиц.

Примеры:

Действия над одномерным массивом (вектором):

```
x = [11 0 33 0 55]';
```

find(x)	find(x == 0)	find(0 < x & x < 10*pi)
1	2	1
3	4	
5		

Действия над двумерным массивом:

```
M = magic(3)
M =
        8     1     6
        3     5     7
        4     9     2
[i, j, s] = find(M > 6); [i j s]
        1        1        1
        3        2        1
        2        3        1
```

Сопутствующие функции: ISEMPTY, SPARSE, NONZEROS, <, <=, >, >=, ==, ~=.

LENGTH **Определение длины вектора**

Синтаксис:

length(x)

Описание:

Функция length(x) возвращает количество элементов вектора x.

Тот же результат можно получить с помощью команды max(size(x)).

Сопутствующие функции: HTML-справка.

SIZE **Определение размеров массива**

Синтаксис:

d = size(X) m = size(X, 1)

[m, n] = size(X) n = size(X, 2)

Описание:

Функция d = size(X) для массива X размером m×n возвращает вектор-строку d = [m n], которая соответственно указывает число строк и столбцов.

Функция [m, n] = size(X) возвращает число строк и столбцов в виде двух переменных.

Функция m = size(X, 1) возвращает только число строк.

Функция n = size(X, 2) возвращает только число столбцов.

Сопутствующие функции: LENGTH.

CROSS **Векторное произведение**

Синтаксис:

c = cross(a, b)

Описание:

Функция c = cross(a, b) формирует векторное произведение двух векторов в трехмерном пространстве.

Результирующий вектор имеет следующее описание:

$\mathbf{c} = \mathbf{a} \times \mathbf{b} = (a_y b_z - a_z b_y)\mathbf{i} + (a_z b_x - a_x b_z)\mathbf{j} + (a_x b_y - a_y b_x)\mathbf{k}$.

Сопутствующие функции: LINSPACE, KRON.

KRON **Формирование тензорного произведения**

Синтаксис:

K = kron(X, Y)

Описание:

Функция K = kron(X, Y) формирует тензорное произведение (произведение Кронекера) двух числовых массивов, так что результирующий массив имеет вид

$$K = X \times Y = [x_{ij} * Y] = \begin{bmatrix} x_{11}Y & x_{12}Y & \dots & x_{1n}Y \\ x_{21}Y & x_{22}Y & \dots & x_{2n}Y \\ \dots & \dots & \dots & \dots \\ x_{m1}Y & x_{m2}Y & \dots & x_{mn}Y \end{bmatrix}.$$

Его размер равен m * size(Y, 1) × n * size(Y, 2).

Сопутствующие функции: CROSS, LINSPACE, MESHGRID.

LINSPACE **Формирование линейного массива равноотстоящих узлов**

Синтаксис:

```
x = linspace(x1, x2)
x = linspace(x1, x2, n)
```

Описание:

Функция x = linspace(x1, x2) формирует линейный массив размера 1×100, начальным и конечным элементами которого являются точки x1 и x2.

Функция x = linspace(x1, x2, n) формирует линейный массив размера 1×n, начальным и конечным элементами которого являются точки x1 и x2.

Сопутствующие функции: LOGSPACE, MESHGRID, :.

LOGSPACE **Формирование узлов логарифмической сетки**

Синтаксис:

```
x = logspace(d1, d2)
x = logspace(d1, d2, n)
```

Описание:

Функция x = logspace(d1, d2) формирует вектор-строку, содержащую 50 равноотстоящих в логарифмическом масштабе точек, которые покрывают диапазон от 10^{d1} до 10^{d2}.

Функция x = logspace(d1, d2, n) формирует вектор-строку, содержащую n равноотстоящих в логарифмическом масштабе точек, которые покрывают диапазон от 10^{d1} до 10^{d2}.

Сопутствующие функции: LINSPACE, MESHGRID, :.

MESHGRID **Формирование узлов двумерной и трехмерной сеток**

Синтаксис:

```
[X, Y] = meshgrid(x, y)
[X, Y] = meshgrid(x)
[X, Y, Z] = meshgrid(x, y, z)
```

Описание:

Функция [X, Y] = meshgrid(x, y) формирует массивы X и Y, которые определяют координаты узлов прямоугольника, задаваемого векторами x и y.

Этот прямоугольник задает область определения функции от двух переменных, которую можно построить в виде 3D-поверхности.

Функция [X, Y] = meshgrid(x) является сокращенной формой записи функции [X, Y] = meshgrid(x, x).

Функция [X, Y, Z] = meshgrid(x, y, z) формирует массивы X, Y и Z, которые определяют координаты узлов параллелепипеда, задаваемого векторами x, y и z. Этот параллелепипед задает область определения для вычисления функции от трех переменных и построения 3D-параметрических поверхностей.

Пример:

Построить двумерную сетку с шагом 0.5 в области -1 < x < 1, -1 < y < 1.

```
[X, Y] = meshgrid(-1 : .5 : 1, -1 : .5 : 1)
X =
     -1     -0.5     0     0.5     1
     -1     -0.5     0     0.5     1
     -1     -0.5     0     0.5     1
     -1     -0.5     0     0.5     1
     -1     -0.5     0     0.5     1
Y =
     -1     -1     -1     -1     -1
   -0.5   -0.5   -0.5   -0.5   -0.5
      0      0      0      0      0
    0.5    0.5    0.5    0.5    0.5
      1      1      1      1      1
```

Сопутствующие функции: LINSPACE, LOGSPACE, SURF, SLICE, :.

Двумерные массивы специального вида

ZEROS **Формирование массива нулей**

Синтаксис:

Y = zeros(n)
Y = zeros(m, n)
Y = zeros(size(A))

Описание:

Функция Y = zeros(n) формирует массив нулей размера n×n.

Функция Y = zeros(m, n) формирует массив нулей размера m×n.

Функция Y = zeros(size(A)) формирует массив нулей, соразмерный с массивом A.

Примеры:

Формирование одномерного массива из 1000 элементов можно выполнить двумя способами:

- в виде цикла с n = 1000
  for i = 1 : n, x(i) = 0; end ,
  что требует для реализации около 1.05 с на PC AT/486 (50 МГц);
- в виде оператора присваивания
  x = zeros(1, 1000);
  что требует для реализации лишь 0.11 с на том же компьютере.

Сопутствующие функции: ONES, EYE, RAND, RANDN.

ONES **Формирование массива единиц**

Синтаксис:

Y = ones(n)
Y = ones(m, n)
Y = ones(size(A))

Описание:

Функция Y = ones(n) формирует массив единиц размера n×n.

Функция Y = ones(m, n) формирует массив единиц размера m×n.

Функция Y = ones(size(A)) формирует массив единиц, соразмерный с массивом A.

Сопутствующие функции: ZEROS, EYE, RAND, RANDN.

RAND **Формирование массива элементов, распределенных по равномерному закону**

Синтаксис:

X = rand(n) rand
X = rand(m, n) rand('state')
X = rand(size(A))

Описание:

Функция X = rand(n) формирует массив размера n×n, элементами которого являются случайные величины, распределенные по равномерному закону в интервале (0, 1).

Функция X = rand(m, n) формирует массив размера m×n, элементами которого являются случайные величины, распределенные по равномерному закону в интервале (0, 1).

Функция X = rand(size(A)) формирует массив, соразмерный с матрицей A, элементами которого являются случайные величины, распределенные по равномерному закону в интервале (0, 1).

Функция rand без аргументов формирует одно случайное число, подчиняющееся равномерному закону распределения в интервале (0, 1), которое изменяется при каждом последующем вызове.

Функция s = rand('state') возвращает вектор из 35 элементов, соответствующий текущему состоянию генератора случайных чисел. Для изменения состояния генератора можно использовать следующие функции:

Функция	*Назначение*
rand('state', s)	Установить генератор в состояние s
rand('state', 0)	Установить генератор в начальное состояние
rand('state', j)	Возвратить генератор к состоянию j
rand('state',sum(100*clock))	Устанавливать в разное состояние на каждом шаге

Замечание:

В системе MATLAB 5 новый генератор случайных чисел, который может формировать случайные числа с плавающей точкой на интервале от 2^{-53} до $1-2^{-53}$. Теоретический период этого генератора равен 2^{1492} значениям. В отличие от версии MATLAB 4, где используется генератор с единственной базой, генератор в версии MATLAB 5 имеет изменяющуюся базу. Функции rand('seed'), rand('seed', 0) и rand('seed', j) позволяют вернуться к использованию генератора версии MATLAB 4.

Примеры:

```
X = rand(3, 4)
X =
        0.2190  0.6793  0.5194  0.0535
        0.0470  0.9347  0.8310  0.5297
        0.6789  0.3835  0.0346  0.6711
```

Сопутствующие функции: RANDN, RANDPERM, SPRAND, SPRANDN.

RANDN

Формирование массива элементов, распределенных по нормальному закону

Синтаксис:

```
X = randn(n)                randn
X = randn(m, n)             randn('state')
X = randn(size(A))
```

Описание:

Функция X = randn(n) формирует массив размера n×n, элементами которого являются случайные величины, распределенные по нормальному закону с математическим ожиданием 0 и среднеквадратическим отклонением 1.

Функция X = randn(m, n) формирует массив размера m×n, элементами которого являются случайные величины, распределенные по нормальному закону с математическим ожиданием 0 и среднеквадратическим отклонением 1.

Функция X = randn(size(A)) формирует массив, соразмерный с матрицей A, элементами которого являются случайные величины, распределенные по нормальному закону с математическим ожиданием 0 и среднеквадратическим отклонением 1.

Команда randn без аргументов формирует одно случайное число, распределенное по нормальному закону с математическим ожиданием 0 и среднеквадратическим отклонением 1, которое изменяется при каждом последующем вызове.

Функция s = randn('state') возвращает вектор из двух элементов, соответствующий текущему состоянию генератора случайных чисел. Для изменения состояния генератора можно использовать следующие функции:

Функция	*Назначение*
randn('state', s)	Установить генератор в состояние s
randn('state', 0)	Установить генератор в начальное состояние
randn('state', j)	Возвратить генератор к состоянию j
randn('state',sum(100*clock))	Устанавливать в разное состояние на каждом шаге

Замечание:

В системе MATLAB 5 новый генератор случайных чисел, который может формировать случайные числа с плавающей точкой на интервале от 2^{-53} до $1-2^{-53}$. Теоретический период этого генератора равен 2^{1492} значениям. В отличие от версии MATLAB 4, где используется генератор с единственной базой, генератор в версии MATLAB 5 имеет изменяющуюся базу. Функции randn('seed'), randn('seed', 0) и randn('seed', j) позволяют вернуться к использованию генератора версии MATLAB 4.

Примеры:

```
X = randn(3, 4)
X =
        1.1650  0.3516  0.0591  0.8717
        0.6268  -0.6965 1.7971  -1.4462
        0.0751  1.6961  0.2641  -0.7012
```

Сопутствующие функции: RAND.

Многомерные массивы

Многомерные массивы - это новый тип массивов системы MATLAB, количество измерений (размерность) которого более двух. Многомерные массивы широко используются при описании страниц двумерных и многомерных данных. Эти массивы могут быть числовыми, символьными, массивами ячеек и массивами структур.

MATLAB поддерживает следующие функции при работе с многомерными массивами:

Функция	*Назначение*
:	Получить доступ к подблокам многомерного массива
size	Определить размер многомерного массива
length	Определить максимальный размер размерности
end	Определить последний индекс размерности
ind2sub	Преобразовать последовательную нумерацию в многомерную
sub2ind	Преобразовать многомерную нумерацию в последовательную
reshape	Преобразовать размеры многомерного массива
repmat	Сформировать многомерный массив из блоков
cat	Сформировать многомерный массив
ndims	Определить размерность многомерного массива
ndgrid	Сгенерировать сетку для многомерной функции
permute, ipermute	Переставить размерности
shiftdim	Изменить размерность массива
flipdim	Отразить многомерный массив относительно заданной размерности
squeeze	Удалить одну из размерностей

Пользователь может расширить состав этих функций, создавая специальные М-файлы для обработки конкретных данных.

: **Формирование многомерного массива**

Синтаксис:

A(i1:i2, j1:j2, k1:k2, ...)
A(n1:n2)
A(:)

Описание:

Начиная с версии MATLAB 5.0 оператор : может быть применен для формирования многомерных массивов или выделения из них подмассивов.

A(:, :, k) - *k-я страница трехмерного массива* A;

A(:, :, :, l) - *l-й трехмерный подмассив четырехмерного массива* A;

A(i, j, k, :) - *одномерный подмассив четырехмерного массива* A;

A(n1:n2) - *последовательность элементов от* n1 *до* n2;

A(:) - *все элементы массива* A, *свернутые в столбец.*

Для формирования многомерных массивов эти конструкции должны использоваться в левой части, а для выделения подмассивов - в правой части оператора присваивания.

Примеры:

A(:, :, 2) = pascal(3)

```
A(:, :, 1) =
   0   0   0
   0   0   0
   0   0   0
A(:, :, 2) =
   1   1   1
   1   2   3
   1   3   6
```

Сопутствующие функции: CAT, RESHAPE.

SIZE **Определение размеров массива**

Синтаксис:

```
d = size(X)
m = size(X, dim)
[d1,d2,...,dk] = size(X)
```

Описание:

Функция d = size(X) для многомерного массива X возвращает вектор-строку длиной ndims(X), каждый элемент которой указывает размер массива по соответствующей размерности.

Функция m = size(X, dim) возвращает размер массива по размерности dim.

Функция [d1, d2, ..., dk] = size(X) возвращает размеры массива по размерностям в виде отдельных переменных. Если количество выходных переменных k равно размерности массива ndims(X), то каждая переменная содержит размер по соответствующей размерности. Если количество выходных переменных k не равно размерности массива ndims(X), то возможно два варианта:

k > ndims(X)	Для всех размерностей, больших ndims(X), возвращаются единицы.
k < ndims(X)	Выходная переменная dk равна произведению размеров всех размерностей от dk до ndims(X).

Пример:

Определить размер массива rand(2, 3, 4):

```
d = size(rand(2, 3, 4))
d = 2 3 4
```

Определить размер массива rand(2, 3, 4) по размерности 2:

```
m = size(rand(2,3,4),2)
m = 3
```

Записать размер по каждой размерности в виде отдельной переменной:

```
[d1, d2, d3] = size(rand(2, 3, 4))
d1 = 2
d2 = 3
d3 = 4
```

Если количество выходных переменных уменьшить до двух:
[d1, d2] = size(rand(2, 3, 4))
d1 = 2
d2 = 12
Если количество выходных переменных увеличить до пяти:
[d1, d2, d3, d4, d5] = size(rand(2, 3, 4))
d1 = 2
d2 = 3
d3 = 4
d4 = 1
d5 = 1

Сопутствующие функции: LENGTH, NDIMS.

LENGTH **Определение максимального размера размерности**

Синтаксис:

n = length(X)

Описание:

Функция n = length(X) эквивалентна функции max(size(X)) для непустых массивов и равна нулю для пустых массивов. В том случае, когда X вектор, результатом является длина вектора.

Примеры:

Случай непустого массива:
X = rand(2,10, 3); n = length(X)
n = 10
Случай пустого массива:
X = rand(2,10, 0); n = length(X)
n = 0

Сопутствующие функции: NDIMS, SIZE.

END **Последний индекс по указанной размерности**

Синтаксис:

B = A(<*индекс*>:end, <*индекс*>)

Описание:

Команда end используется в индексных выражениях операторов вида B = A(<*индекс*>:end, <*индекс*>) для задания последнего индекса по указанной размерности. В этом смысле значение end равно size(A, k).

Если команда end используется для расширения массива, например, в форме A = (end+1, :), необходимо убедиться, что массив A уже существует.

Примеры:

Сформируем массив

```
A = rand(5, 4)
A =
   0.9501   0.7621   0.6154   0.4057
   0.2311   0.4565   0.7919   0.9355
   0.6068   0.0185   0.9218   0.9169
   0.4860   0.8214   0.7382   0.4103
   0.8913   0.4447   0.1763   0.8936
```

и выделим из него фрагмент последней строки

```
B = A(end, 2:end)
B =   0.4447   0.1763   0.8936
```

Дополним матрицу A единичным диагональным элементом

```
A(end+1, end+1) = 1
A =
   0.9501   0.7621   0.6154   0.4057        0
   0.2311   0.4565   0.7919   0.9355        0
   0.6068   0.0185   0.9218   0.9169        0
   0.4860   0.8214   0.7382   0.4103        0
   0.8913   0.4447   0.1763   0.8936        0
        0        0        0        0   1.0000
```

Сопутствующие функции: SIZE.

IND2SUB **Преобразовать последовательную нумерацию в многомерную**

Синтаксис:

```
[I, J] = ind2sub(siz ,IND)
[I1,I2,I3,...,In] = ind2sub(siz, IND)
```

Описание:

Функция [I, J] = ind2sub(siz, IND) возвращает индексные массивы I и J, эквивалентные массиву последовательной нумерации IND, который поставлен в соответствие некоторому массиву размера siz. Например, для двумерных массивов конструкция [I, J] = ind2sub(size(A), find(A>5)) равносильна [I, J] = find(A>5).

Функция [I1,I2,I3,...,In] = ind2sub(siz, IND) возвращает n индексных массивов I1, I2, ..., In, эквивалентных массиву последовательной нумерации IND, который поставлен в соответствие некоторому массиву размера siz.

Примеры:

Преобразовать последовательную нумерацию трехмерного массива размера 2×2×2 в многомерную:

```
A= rand(2, 2, 2);
[I1, I2, I3] = ind2sub(size(A), find(A>0));
[I1 I2 I3]
```

```
ans =
     1     1     1
     2     1     1
     1     2     1
     2     2     1
     1     1     2
     2     1     2
     1     2     2
     2     2     2
```

Последовательная нумерация в данном случае задается условием find(A>0). Реализованное преобразование поясняется следующей схемой.

Последовательная нумерация

5	7
6	8

1	3
2	4

Многомерная нумерация

1,1,2	1,2,2
2,1,2	2,2,2

1,1,1	1,2,1
2,1,1	2,2,1

Сопутствующие функции: SUB2IND, FIND.

SUB2IND **Преобразовать многомерную нумерацию в последовательную**

Синтаксис:

IND = sub2ind(siz, I, J)
IND = sub2ind(siz, I1,I2,...,In)

Описание:

Функция IND = sub2ind(siz, I, J) возвращает массив последовательной нумерации IND, эквивалентный индексным массивам многомерной нумерации I и J, которые поставлены в соответствие некоторому массиву размера siz.

Функция IND = sub2ind(siz, I1,I2,...,In) возвращает массив последовательной нумерации IND, эквивалентный n индексным массивам многомерной нумерации I1, I2, ..., In, которые поставлены в соответствие некоторому массиву размера siz.

Примеры:

Преобразовать многомерную нумерацию трехмерного массива размера 2×2×2 в последовательную:

```
A= rand(2, 2, 2);
[I1, I2, I3] = ind2sub(size(A), find(A>0));
[I1 I2 I3]
```

```
ans =
   1   1   1
   2   1   1
   1   2   1
   2   2   1
   1   1   2
   2   1   2
   1   2   2
   2   2   2
IND = sub2ind(size(A), I1, I2, I3)
IND =
   1
   2
   3
   4
   5
   6
   7
   8
```

Реализованное преобразование поясняется следующей схемой.

Многомерная нумерация

1,1,2	1,2,2
2,1,2	2,2,2

1,1,1	1,2,1
2,1,1	2,2,1

Последовательная нумерация

5	7
6	8

1	3
2	4

Сопутствующие функции: SUB2IND, FIND.

RESHAPE **Преобразование размеров многомерного массива**

Синтаксис:

B = reshape(A, m, n, p, ...)

Описание:

Функция B = reshape(A, m, n, p, ...) возвращает многомерный массив размера m×n×p× ..., сформированный из элементов массива A. Произведение m * n* p*... должно быть равно prod(size(A)).

Конструкция reshape(A, [m n p ...]) равносильна конструкции reshape(A, m, n, p, ...).

Пример:

Пусть

A = ones(3, 4, 2)

```
A(:, :, 1) =
   1   1   1   1
   1   1   1   1
   1   1   1   1
A(:, :, 2) =
   1   1   1   1
   1   1   1   1
   1   1   1   1
```

Тогда

```
B = reshape(A, [3 2 2 2])
B(:, :, 1, 1) =
   1   1
   1   1
   1   1
B(:, :, 1, 2) =
   1   1
   1   1
   1   1
B(:, :, 2, 1) =
   1   1
   1   1
   1   1
B(:, :, 2, 2) =
   1   1
   1   1
   1   1
```

Сопутствующие функции и операторы: SQUEEZE, SHIFTDIM, :.

REPMAT **Формирование многомерного массива из блоков**

Синтаксис:

B = repmat(A, [m n p ...])

Описание:

Функция B = repmat(A, [m n p ...]) формирует многомерный массив, сформированный из m×n ×p × ... блоков A.

Конструкция repmat(A, m, n, p, ...) равносильна конструкции repmat(A, [m n p ...]).

В том случае, когда A - скаляр, использование функции repmat(A, [m n p ...]) может оказаться предпочтительнее по времени выполнения, чем конструкция A*ones(m, n, p, ...).

Сопутствующие функции и операторы: MESHGRID.

CAT **Объединение массивов**

Синтаксис:

C = cat(dim, A, B)

C = cat(dim, A1, A2, A3, A4 ...)

Описание:

Функция C = cat(dim, A, B) объединяет массивы A и B вдоль размерности dim.

Функция C = cat(dim, A1, A2, A3, A4 ...) объединяет множество исходных массивов Ai вдоль размерности dim.

При этом cat(1, A, B) равносильно массиву [A; B], объединяемому вдоль строк; cat(2, A, B) равносильно массиву [A B], объдиняемому вдоль столбцов.

Функции вида cat(dim, A{:}) и cat(dim, A.<*имя_поля*>) задают объединение массива ячеек или массива записей, содержащего числовые матрицы, в некоторый многомерный массив.

Пример:

Пусть заданы два двумерных массива A и B:

```
A =                    A =
     1     2                5     6
     3     4                7     8
```

Выполним их объединение вдоль разных размерностей:

C = cat(1, A, B)

1	2
3	4
5	6
7	8

C = cat(3, A, B)

5	6
7	8

C = cat(2, A, B)

1	2	5	6
3	4	7	8

1	2
3	4

Последовательность операторов

```
M = magic(3);
P = pascal(3);
C = cat(4, M, P)
```

создает следующий многомерный массив размера 3×3×1×2:

```
C(:, :, 1, 1) =
     8     1     6
     3     5     7
     4     9     2
C(:, :, 1, 2) =
     1     1     1
     1     2     3
     1     3     6
```

Объединяя массив M и массив P, дополненный единичным столбцом, в массив ячеек S:

```
S = {M [P  ones(size(P, 1), 1)]};
for i=1:length(S),
    siz{i} = size(S{i});
end
sizes = cat(1, siz{:})
```

можно сформировать двумерный массив sizes, элементами которого являются размеры входных массивов

```
sizes =
     3     3
     3     4
```

Сопутствующие функции: NUM2CELL.

Переопределение метода: funfun\inline\cat.m

NDIMS **Количество размерностей многомерного массива**

Синтаксис:

n = ndims(A)

Описание:

Функция n = ndims(A) возвращает количество размерностей многомерного массива A, которое всегда больше или равно 2. Оконечные единичные размерности во внимание не принимаются.

Алгоритм:

ndims(X) = length(size(X))

Пример:

X = rand(1, 2, 10, 1, 3, 1); n = ndims(X)
n = 5

Сопутствующие функции: SIZE.

NDGRID **Генерация сетки для многомерных функций и интерполяции**

Синтаксис:

[X1, X2, X3, ...] = ndgrid(x1, x2, x3, ...)
[X1, X2, ...] = ndgrid(x)

Описание:

Функция [X1, X2, X3, ...] = ndgrid(x1, x2, x3, ...) преобразовывает области, заданные векторами x1, x2, x3, ... в массивы X1, X2, X3, ... которые можно использовать в качестве сетки для вычисления функций нескольких переменных и многомерной интерполяции. При этом i-я размерность выходного массива X_i повторяет элементы вектора x_i.

Функция [X1, X2, ...] = ndgrid(x) равносильна функции [X1, X2, ...] = = ndgrid(x, x, ...).

Пример:

Вычислить функцию от трех переменных x2*exp(-x1^2-x2^2-x^3) на области -2 < x1 < 2, -2 < x2 < 2, -2 < x3 < 2 и построить ее сечения, используя команду slice:

```
[x1, x2, x3] = ndgrid(-2:.2:2, -2:.25:2, -2:.16:2);
z = x2 .* exp(-x1.^2 - x2.^2 - x3.^2);
slice(x2, x1, x3, z, [-1.2 .8 2], 2, [-2 -.2])
```

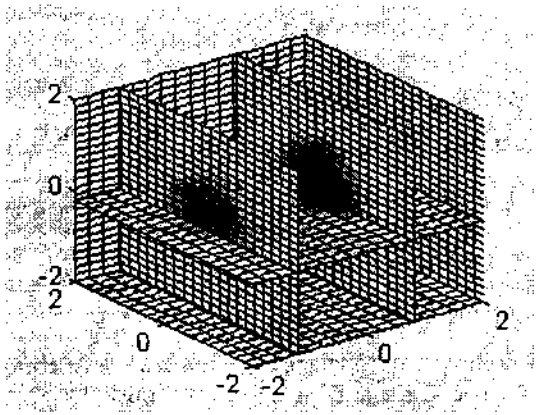


Рис. 3.2

Замечание:

Функция ndgrid аналогична функции meshgrid, за исключением того, что первые два аргумента переставлены местами, то есть функция [X1, X2, X3] = ndgrid(x1, x2, x3) дает тот же результат, что и функция [X2, X1, X3] = meshgrid(x2, x1, x3).

В силу этого обстоятельства функция ndgrid лучше подходит для решения многомерных задач, в то время как функция meshgrid - для решения пространственных задач в двумерном и трехмерном пространствах.

Сопутствующие функции: MESHGRID, INTERPN.

PERMUTE, IPERMUTE

Прямая и обратная перестановки размерностей многомерного массива

Синтаксис:

B = permute(A, <*вектор перестановок*>)
A = ipermute(B, <*вектор перестановок*>)

Описание:

Функция B = permute(A, <*вектор перестановок*>) осуществляет перестановку размерностей многомерного массива A в соответствии с порядком, определенным вектором перестановок. Значения элементов массива оста-

ются неизменными, но порядок размещения последних определяется вектором перестановок; в свою очередь, элементы вектора перестановок - это числа от 1 до N, переставленные соответствующим образом.

Функция A = ipermute(B, <*вектор перестановок*>) осуществляет обратную перестановку размерностей многомерного массива A в соответствии с порядком, определенным вектором перестановок.

Замечание:

Функции permute и ipermute обобщают операцию траспонирования (.') на случай многомерных массивов.

Пример:

```
A = rand(1,2,3,4);
B  = permute(A,  [3 2 1 4]);
size(B)
ans =     3     2     1     4
```

Массив с переставленными размерностями имеет размер 3×2×1×4.

```
C = ipermute(B, [3 2 1 4]);
isequal(A, C)
ans =     1
```

Таким образом, массивы C и A идентичны.

Сопутствующие функции: нет.

SHIFTDIM **Сдвиг размерностей многомерного массива**

Синтаксис:

B = shiftdim(X, n)
[B, n] = shiftdim(X)

Описание:

Функция B = shiftdim(X, n) сдвигает n размерностей многомерного массива X; если n положительное число, выполняется сдвиг на n размерностей влево, а n первых размерностей подставляются в конец (круговая перестановка); если n отрицательное число, выполняется сдвиг на n размерностей вправо, а n первых размерностей дополняются единичными.

Функция [B, n] = shiftdim(X) возвращает с тем же количеством элементов, что и X, но с удаленными ведущими единичными размерностями; количество удаленных размерностей фиксируется переменной n.

Функция shiftdim удобна тем, что, подобно функциям sum и diff, работает с первой неединичной размерностью.

Замечание:

Если X - скаляр, то функция shiftdim(X) не выполняется.

Пример:

```
A = rand(1, 1, 3, 1, 2);
[B, n] = shiftdim(A)
B(:, :, 1) =
    0.3046
    0.1897
    0.1934
B(:, :, 2) =
    0.6822
    0.3028
    0.5417
n =     2
size(B)
ans =     3     1     2
```

Массив B имеет размер 3×1×2 и n = 2.

```
C = shiftdim(B, -n);
isequal(A, C)
ans =     1
```

Таким образом, массивы C и A идентичны.

```
D = shiftdim(A, 3);
size(D)
ans =     1     2     1     1     3
```

Сопутствующие функции: RESHAPE, SQUEEZE.

FLIPDIM **Отражение многомерного массива относительно указанной размерности**

Синтаксис:

B = flipdim(A, dim)

Описание:

Функция B = flipdim(A, dim) выполняет отражение массива относительно размерности dim. Если dim равно 1, то массив отражается относительно горизонтальной оси, если dim равно 2, то массив отражается относительно вертикальной оси. Таким образом, оператор flipdim(A, 1) равносилен flipud(A), а оператор flipdim(A, 2) соответствует fliplr(A).

Пример:

```
A = rand(2, 2, 2)
A(:, :, 1) =
   0.9355   0.4103
   0.9169   0.8936
A(:, :, 2) =
   0.0579   0.8132
   0.3529   0.0099
```

```
flipdim(A, 3)
ans(:,:,1) =
   0.0579   0.8132
   0.3529   0.0099
ans(:, :, 2) =
   0.9355   0.4103
   0.9169   0.8936
```

Сопутствующие функции: FLIPLR, FLIPUD, ROT90, PERMUTE.

SQUEEZE **Удаление всех единичных размерностей многомерного массива**

Синтаксис:

B = squeeze(A)

Описание:

Функция B = squeeze(A) возвращает массив B с теми же элементами, с которыми она возвращает и массив A, но в котором удалены размерности, равные 1.

Пример:

Рассмотрим трехмерный массив A=rand(2, 1, 3) размера 2×1×3. Этот массив имеет размерность столбца, равную 1, то есть на каждой странице размещен один вектор-столбец:

```
A = rand(2, 1, 3)
A(:, :, 1) =
    0.9218
    0.7382
A(:, :, 2) =
    0.1763
    0.4057
A(:, :, 3) =
    0.9355
    0.9169
```

Применение функции squeeze превращает его в двумерный массив размера 2×3:

```
squeeze(A)
ans =
    0.9218    0.1763    0.9355
    0.7382    0.4057    0.9169
```

Сопутствующие функции: RESHAPE, SHIFTDIM.

Многомерные массивы специального вида

ZEROS **Формирование массива нулей**

Синтаксис:

Y = zeros(d1, d2, d3 ...)
Y = zeros([d1 d2 d3 ...])
Y = zeros(size(A))

Описание:

Функции Y = zeros(d1, d2, d3 ...) и Y = zeros([d1 d2 d3 ...]) возвращают массив нулей размера d1 × d2 × d3× ... .

Функция Y = zeros(size(A)) формирует массив нулей, соразмерный с массивом A.

Сопутствующие функции: ONES, EYE, RAND, RANDN.

ONES **Формирование массива единиц**

Синтаксис:

Y = ones(d1, d2, d3 ...)
Y = ones([d1 d2 d3 ...])
Y = ones(size(A))

Описание:

Функции Y = ones(d1, d2, d3 ...) и Y = ones([d1 d2 d3 ...]) возвращают массив единиц размера d1 × d2 × d3× ... .

Функция Y = ones(size(A)) формирует массив единиц, соразмерный с массивом A.

Сопутствующие функции: ZEROS, EYE, RAND, RANDN.

RAND **Формирование массива элементов, распределенных по равномерному закону**

Синтаксис:

Y = rand((d1, d2, d3 ...)
Y = rand(([d1 d2 d3 ...])
Y = rand((size(A))

Описание:

Функции Y = rand(d1, d2, d3 ...) и Y = rand([d1 d2 d3 ...]) возвращают массив размера d1×d2×d3× ..., элементами которого являются случайные величины, распределенные по равномерному закону в интервале (0, 1).

Функция X = rand(size(A)) формирует массив, соразмерный с матрицей A, элементами которого являются случайные величины, распределенные по равномерному закону в интервале (0, 1).

Сопутствующие функции: RANDN, RANDPERM, SPRAND, SPRANDN.

RANDN **Формирование массива элементов, распределенных по нормальному закону**

Синтаксис:

Y = randn((d1, d2, d3 ...)
Y = randn(([d1 d2 d3 ...])
Y = randn((size(A))

Описание:

Функции Y = randn(d1, d2, d3 ...) и Y = randn([d1 d2 d3 ...]) возвращают массив размера d1 × d2 × d3× ..., элементами которого являются случайные величины, распределенные по нормальному закону с математическим ожиданием 0 и среднеквадратическим отклонением 1.

Функция X = randn(size(A)) формирует массив, соразмерный с матрицей A, элементами которого являются случайные величины, распределенные по нормальному закону с математическим ожиданием 0 и среднеквадратическим отклонением 1.

Сопутствующие функции: RAND.

Функции для работы с массивами записей

Массивы записей - это новый тип массивов в системе MATLAB, в котором разрешается накапливать в виде записей разнородные данные. Отличительная особенность таких массивов - наличие именованных полей.

MATLAB поддерживает следующие функции при работе с массивами записей:

Функция	*Описание*
struct	Создать массив записей
fieldnames	Получить имена полей
getfield	Получить содержимое поля
setfield	Установить содержимое поля
rmfield	Удалить поле
isfield	Истинно, если это поле массива записей
isstruct	Истинно, если это массив записей

Пользователь может расширить состав функций, создавая специальные М-файлы для обработки конкретных данных.

STRUCT **Создать массив записей (структуру)**

Синтаксис:

S = struct('*<имя_поля1>*', *<значение>*, '*<имя_поля2>*', *<значение>*, ...)

Описание:

Функция S = struct('*<имя_поля1>*', *<значение>*, '*<имя_поля2>*', *<значение>*, ...) создает массив записей (структуру) с заданными именами и значениями полей.

Пример:

Воспользуемся функцией struct, чтобы создать структуру patient размера 1×1:

```
patient = struct('name',' John Doe', 'billing', 127.00,...
'test', [79 75 73; 180 178 177.5; 220 210 205])
patient =
        name: ' John Doe'
        billing: 127
           test: [3x3 double
```

Сопутствующие функции: CLASS, CELL, GETFIELD, SETFIELD, RMFIELD, FIELDNAMES.

FIELDNAMES **Получить имена полей**

Синтаксис:

names = fieldnames(S)

Описание:

Функция names = fieldnames(S) возвращает имена полей структуры S в виде строк массива ячеек.

Пример:

Задана следующая структура A размера 1×2:

A(1).data = [3 4 7; 8 0 1];
A(1).nest.testnum = 'Test 1';
A(1).nest.xdata = [4 2 8];
A(1).nest.ydata = [7 1 6];
A(2).data = [9 3 2; 7 6 5];
A(2).nest.testnum = 'Test 2';
A(2).nest.xdata = [3 4 2];
A(2).nest.ydata = [5 0 9]

Определим имена ее полей, используя функцию fieldnames:

```
fieldnames(A)
ans =
    'data'
    'nest'
```

Сопутствующие функции: GETFIELD, SETFIELD.

GETFIELD **Получить содержимое поля**

Синтаксис:

F = getfield(s, '<*имя_поля*>')
F = getfield(S, {i, j}, '<*имя_поля*>', {k})

Описание:

Функция F = getfield(s, '<*имя_поля*>'), где элемент структуры или структура s должны иметь размер s, возвращает содержимое указанного поля.

Функция F = getfield(S, {i, j}, '<*имя_поля*>', {k}) равносильна следующему оператору присваивания F = S(i, j).<*имя_поля*>(k). Все индексы передаются как массивы ячеек и заключаются в фигурные скобки; имена полей передаются как строки.

Пример:

Задана следующая структура A размера 1×2:

```
A(1).data = [3 4 7; 8 0 1];
A(1).nest.testnum = 'Test 1';
A(1).nest.xdata = [4 2 8];
A(1).nest.ydata = [7 1 6];
A(2).data = [9 3 2; 7 6 5];
A(2).nest.testnum = 'Test 2';
A(2).nest.xdata = [3 4 2];
A(2).nest.ydata = [5 0 9]
```

Определим содержимое поля A(1).nest:

```
getfield(A(1), 'nest')
```

```
ans =
    testnum: 'Test 1'
      xdata: [4 2 8]
      ydata: [7 1 6]
```

Это также равносильно следующему оператору:

```
getfield(A, {1}, 'nest')
ans =
    testnum: 'Test 1'
      xdata: [4 2 8]
      ydata: [7 1 6]
```

Сравните эти результаты с обращением к оператору A.nest:

```
A.nest
ans =
    testnum: 'Test 1'
      xdata: [4 2 8]
      ydata: [7 1 6]
ans =
    testnum: 'Test 2'
      xdata: [3 4 2]
      ydata: [5 0 9]
```

Сопутствующие функции: SETFIELD, FIELDNAMES.

SETFIELD **Установить содержимое поля**

Синтаксис:

s = setfield(s, '<*имя_поля*>', V)

s = setfield(S, {i, j}, '<*имя_поля*>', {k}, V)

Описание:

Функция s = setfield(s, '<*имя_поля*>', V), где элемент структуры или структура s должны иметь размер 1×1, присваивает указанному полю значение V.

Функция s = setfield(S, {i, j}, '<*имя_поля*>', {k}, V) равносильна следующему оператору присваивания S(i, j).<*имя_поля*>(k) = V. Все индексы передаются как массивы ячеек и заключаются в фигурные скобки; имена полей передаются как строки.

Пример:

Задана следующая структура A размера 1×2:

A(1).data = [3 4 7; 8 0 1];

```
A(1).nest.testnum = 'Test 1';
A(1).nest.xdata = [4 2 8];
A(1).nest.ydata = [7 1 6];
A(2).data = [9 3 2; 7 6 5];
A(2).nest.testnum = 'Test 2';
A(2).nest.xdata = [3 4 2];
A(2).nest.ydata = [5 0 9]
```

Присвоить новое значение полю A(1).nest.xdata:

```
A = setfield(A(1), 'nest.xdata', [5 3 9]);
getfield(A(1), 'nest')
ans =
    testnum: 'Test 1'
      xdata: [5 3 9]
      ydata: [7 1 6]
```

Это также равносильно следующему оператору:

```
A = setfield(A, {1}, 'nest.xdata', [5 3 9]);
getfield(A, {1}, 'nest.xdata')
ans =
    testnum: 'Test 1'
      xdata: [5 3 9]
      ydata: [7 1 6]
```

Сопутствующие функции: GETFIELD, FIELDNAMES.

RMFIELD **Удалить поле**

Синтаксис:

S = rmfield(S, '<*имя_поля*>')
S = rmfield(S, F)

Описание:

Функция S = rmfield(S, '<*имя_поля*>') удаляет указанное поле из структуры.

Функция S = rmfield(S, F), где F - символьный массив имен полей или массив ячеек соответствующих строк, удаляет все указанные поля из структуры.

Замечание:

Удалить таким способом все поля из структуры нельзя.

Пример:

Задана следующая структура A размера 1×2:

```
A(1).data = [3 4 7; 8 0 1];
A(1).nest.testnum = 'Test 1';
A(1).nest.xdata = [4 2 8];
A(1).nest.ydata = [7 1 6];
A(2).data = [9 3 2; 7 6 5];
A(2).nest.testnum = 'Test 2';
A(2).nest.xdata = [3 4 2];
A(2).nest.ydata = [5 0 9]
```

Удалить A(1).data:

```
B=rmfield(A,'data')
B =
1x2 struct array with fields:
    nest
B.nest
```

```
ans =
    testnum: 'Test 1'
      xdata: [4 2 8]
      ydata: [7 1 6]
ans =
    testnum: 'Test 2'
      xdata: [3 4 2]
      ydata: [5 0 9]
```

Попытка удалить поле nest приводит к сообщению об ошибке:

```
B = rmfield(B, 'nest')
   ??? To RESHAPE the number of elements must not change.
   Для выполнения функции RESHAPE должно быть изменено
   количество элементов
   Error in ==>
   d:\matlab5\toolbox\matlab\datatypes\rmfield.m
   On line 43  ==> t = reshape(t,size(s));
   Ошибка в ==>
   d:\matlab5\toolbox\matlab\datatypes\rmfield.m
   В строке 43 ==> t = reshape(t,size(s));
```

Сопутствующие функции: SETFIELD, GETFIELD, FIELDNAMES, STRVCAT.

ISFIELD **Логическая проверка поля**

Синтаксис:

k = isfield(S, '<*имя_поля*>')

Описание:

Функция k = isfield(S, '<*имя_поля*>') возвращает 1 (логическое TRUE), если указанное имя действительно является именем поля данной структуры.

Пример:

Задана следующая структура A размера 1×2:

```
A(1).data = [3 4 7; 8 0 1];
A(1).nest.testnum = 'Test 1';
A(1).nest.xdata = [4 2 8];
A(1).nest.ydata = [7 1 6];
A(2).data = [9 3 2; 7 6 5];
A(2).nest.testnum = 'Test 2';
A(2).nest.xdata = [3 4 2];
A(2).nest.ydata = [5 0 9]
```

Проверить, являются ли поля 'data', 'nest', 'nest.xdata' полями структуры A.

```
isfield(A,'data')
ans =      1
isfield(A,'nest')
ans =     1
isfield(A,'nest.xdata')
ans =     0
```

Сопутствующие функции: SETFIELD, GETFIELD, FIELDNAMES.

ISSTRUCT **Логическая проверка структуры**

Синтаксис:

k = isstruct(S)

Описание:

Функция k = isstruct(S) возвращает 1 (логическое TRUE), если указанное имя действительно является именем структуры, и 0 - в противном случае.

Пример:

Задана следующая структура A размера 1×2:

```
A(1).data = [3 4 7; 8 0 1];
A(1).nest.testnum = 'Test 1';
A(1).nest.xdata = [4 2 8];
A(1).nest.ydata = [7 1 6];
A(2).data = [9 3 2; 7 6 5];
A(2).nest.testnum = 'Test 2';
A(2).nest.xdata = [3 4 2];
A(2).nest.ydata = [5 0 9]
```

Проверить, является ли объект A структурой.

```
isstruct(A)
ans =      1
```

Сопутствующие функции: STRUCT, ISCELL, ISNUMERIC, ISOBJECT.

Функции и команды обработки массивов ячеек

В систему MATLAB 5 впервые включен специальный тип массивов ячеек, элементы которого сами, в свою очередь, являются массивами. Поддержаны следующие функции при работе с массивами ячеек:

Функция	*Описание*
cell	Создать массив ячеек
celldisp	Показать содержимое массива ячеек
cellplot	Показать графическую структуру массива ячеек
num2cell	Преобразовать числовой массив в массив ячеек
deal	Обмен данными между любыми классами массивов
cell2struct	Преобразовать массив ячеек в структуру
struct2cell	Преобразовать структуру в массив ячеек
iscell	Истинно, если это массив ячеек

Пользователь может расширить состав этих функций, создавая специальные М-файлы для обработки конкретных данных.

CELL **Создать массив ячеек**

Синтаксис:

```
c = cell(n)
c = cell(m, n)
c = cell([m n])
c = cell(m, n, p,...)
c = cell([m n p ...])
c = cell(size(A))
```

Описание:

Функция c = cell(n) создает массив ячеек, состоящих из пустых матриц, размера n×n. Сообщение об ошибке возникает в том случае, если n не является скаляром.

Функции c = cell(m, n) и c = cell([m, n]) создают массив ячеек, состоящих из пустых матриц, размера m×n. Аргументы m и n должны быть скалярными.

Функции c = cell(m, n, p,...) и c = cell([m n p ...]) создают многомерный массив ячеек, состоящих из пустых матриц, размера m×n×p× ... . Аргументы m, n, p, ... должны быть скалярными.

Функция c = cell(size(A)) создает массив ячеек, состоящих из пустых матриц, того же размера, что и массив A.

Пример:

```
A = ones(2, 2)
A =
   1   1
   1   1
c = cell(size(A))
c =
   []   []
   []   []
```

Сопутствующие функции: ONES, RAND, RANDN, ZEROS.

CELLDISP **Вывести на экран содержимое массива ячеек**

Синтаксис:

```
celldisp(C)
```

Описание:

Команда celldisp(C) выводит на экран содержимое массива ячеек.

Пример:

Выведем на экран содержимое следующего массива ячеек размера 2×3:

```
C = {[1 2]  'Tony'  3+4i;  [1 2; 3 4]  -5  'abc'};
celldisp(C)
```

```
C{1,1} =      1     2
C{2,1} =
     1     2
     3     4
C{1,2} =Tony
C{2,2} =    -5
C{1,3} =   3.0000+ 4.0000i
C{2,3} =   abc
```

Сопутствующие функции: CELLPLOT.

CELLPLOT **Вывести на экран графическую структуру массива ячеек**

Синтаксис:

cellplot(C)
cellplot(C, 'legend')
handles = cellplot(...)

Описание:

Команда cellplot(C) выводит в графическое окно содержимое массива ячеек C. Закрашенные прямоугольники соответствуют векторам и массивам, скалярные величины и короткие строки символов выводятся в виде текста.

Команда cellplot(C, 'legend') кроме графического изображения выводит описание цветов для различных типов данных.

Функция handles = cellplot(C) выводит в графическое окно содержимое массива ячеек C и возвращает вектор поддержек.

Ограничение:

Команда cellplot может выводить только графическую структуру для двумерных массивов ячеек.

Пример:

Выведем на экран содержимое следующего массива ячеек размера 2×3:

```
c{1, 1} = '2-by-2';
c{1, 2} = 'eigenvalues of eye(2)';
c{2, 1} = eye(2);
c{2, 2} = eig(eye(2));
cellplot(c, 'legend')
```

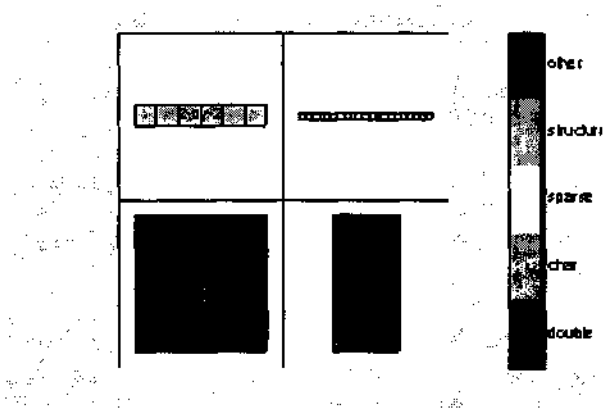


Рис. 3.3

Сопутствующие функции: CELLDISP.

CELLSTR **Преобразовать массив строк в массив символьных ячеек**

Синтаксис:

C = cellstr(S)

Описание:

Функция C = cellstr(S) преобразовывает массив строк S в массив символьных ячеек C.

Пример:

Задан следующий массив строк S размера 3×4:

```
S = ['abc ' ;
        'defg' ;
        'hi  ']
S =
   abc
   defg
   hi
```

Функция C = cellstr(S) возвращает массив ячеек размера 3×1:

```
C = cellstr(S)
C =
    'abc'
    'defg'
    'hi'
```

Сопутствующие функции: ISCELLSTR, STRINGS.

DEAL **Установить соответствие между входами и выходами**

Синтаксис:

[Y1, Y2, Y3,...] = deal(X)
[Y1, Y2, Y3,...] = deal(X1, X2, X3,...)

Описание:

Функция [Y1, Y2, Y3,...] = deal(X) копирует единственный вход на все выходы, реализуя следующее соответствие Y1 = X, Y2 = X, Y3 = X, ...

Функция [Y1, Y2, Y3,...] = deal(X1, X2, X3,...) устанавливает следующее соответствие между входами и выходами Y1 = X1, Y2 = X2, Y3 = X3, ...

Замечание:

Функция deal исключительно полезна при применении в следующих конструкциях при работе с массивами ячеек и массивами записей:

- оператор [S.field] = deal(X) присваивает всем полям структуры S с именем field значение X. Если S не существует, надо использовать оператор [S(1:m).field] = deal(X);
- оператор [X{ : }] = deal(A.field) копирует поля структуры A с именем field в массив ячеек X. Если X не существует, надо использовать оператор [X{1:m}] = deal(A.field);
- оператор [Y1, Y2, Y3, ...] = deal(X{ : }) копирует содержимое массива ячеек X в отдельные переменные Y1, Y2, Y3, ... ;
- оператор [Y1,Y2,Y3,...] = deal(S.field) копирует поля структуры S с именем field в отдельные переменные Y1, Y2, Y3, ...

Пример:

Скопировать содержимое массива ячеек C размера 1×4 в 4 выходные переменные:

```
C = {rand(3) ones(3, 1) eye(3) zeros(3, 1)};
[a, b, c, d] = deal(C{ : })
a =
    0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214
b =
     1
     1
     1
c =
     1     0     0
     0     1     0
     0     0     1
d =
     0
     0
     0
```

Скопировать содержимое всех полей name структуры A размера 1×2 в отдельные переменные:

```
A.name = 'Pat';          A.number = 176554;
A(2).name = 'Tony';  A(2).number = 901325;
[name1, name2] = deal(A(:).name)
name1 = Pat
name2 = Tony
```

Сопутствующие функции: VARARGIN, VARARGOUT, CELL2STRUCT, STRUCT2CELL, NUM2CELL, CAT.

ISCELL **Выявление массива ячеек**

Синтаксис:

k = iscell(C)

Описание:

Функция k = iscell(C) возвращает логическое TRUE (1), если C - массив ячеек, и логическое FALSE (0) - в противном случае.

Сопутствующие функции: функции группы IS*.

NUM2CELL **Преобразовать массив чисел в массив ячеек**

Синтаксис:

C = num2cell(A)
C = num2cell(A, dims)

Описание:

Функция C = num2cell(A) преобразовывает массив A в массив ячеек, размещая каждый элемент массива A в отдельной ячейке. Размер массива ячеек будет совпадать с размерами массива A.

Функция C = num2cell(A, dims) преобразовывает только те элементы массива A в массив ячеек, которые заданы вторым аргументом.

Пример:

Рассмотрим некоторый массив чисел A = rand(3):

```
A = rand(3)
A =
    0.4447    0.9218    0.4057
    0.6154    0.7382    0.9355
    0.7919    0.1763    0.9169
```

и применим к нему следующие преобразования:

```
celldisp(num2cell(A, 2))
ans{1} =     0.4447    0.9218    0.4057
ans{2} =     0.6154    0.7382    0.9355
ans{3} =     0.7919    0.1763    0.9169
```

```
celldisp(num2cell(A, [1 3]))

ans{1} =          ans{2} =          ans{3} =
    0.4447            0.9218            0.4057
    0.6154            0.7382            0.9355
    0.7919            0.1763            0.9169
```

В последнем случае столбцы размещены по отдельным ячейкам.

Сопутствующие функции: CAT.

CELL2STRUCT **Преобразовать массив ячеек в массив записей**

Синтаксис:

S = cell2struct(C, fields, dim)

Описание:

Функция S = cell2struct(C, fields, dim) преобразовывает массив ячеек C в массив записей S вдоль размерности dim, сохраняя размер массива C по этой размерности в записи структуры. Аргумент fields может быть массивом строк или массивом строковых ячеек.

Пример:

Рассмотрим массив ячеек размера 1 по первой размерности и размера 3 по второй

```
c = {'tree', 37.4, 'birch'}
c =
    'tree'    [37.4000]    'birch'
```

и преобразуем его по второй размерности в структуру с полями f = = {'category', 'height', 'name'}:

```
f = {'category', 'height', 'name'};
s = cell2struct(c, f, 2)
s =
    category: 'tree'
      height: 37.4000
        name: 'birch'
```

Сопутствующие функции: FIELDNAMES, STRUCT2CELL.

STRUCT2CELL **Преобразовать массив записей в массив ячеек**

Синтаксис:

C = struct2cell(S)

Описание:

Функция C = struct2cell(S) преобразовывает массив записей S размера m×n (с p полями) в массив ячеек C размера p×m×n.

Если массив записей S многомерный, то массив ячеек C имеет размер [p size(S)].

Пример:

Следующие операторы:

```
clear S,
S.category = 'tree'; S.height = 37.4; S.name = 'birch';
```

создают структуру

```
S =
  category: 'tree'
      height: 37.4000
        name: 'birch
```

Преобразуем эту структуру в массив ячеек:

```
C = struct2cell(S)
C =
    'tree'
    [37.4000]
    'birch'
```

Сопутствующие функции: CELL2STRUCT, FIELDS.

LISTS **Определение списков значений**

Синтаксис:

help lists

Описание:

Команда help lists выводит на экран следующий комментарий к определению и использованию списков значений при работе с массивами записей и массивами ячеек.

Извлечение множественных данных из массивов записей и массивов ячеек осуществляется с помощью *списков значений*.

Список значений для *массива записей* - это объединение одноименных полей S.name = [S(1).name S(2).name ... S(end).name].

Список значений для *массива ячеек* - это объединение ячеек C{:} = [C{1} C{2} ... C{end}].

Конструкции вида S(m:n).name, C{m:n} также представляют собой списки значений.

Списки значений используются в следующих случаях:

- в командной строке для вывода значений на экран - S.name, C{:};
- при вызове М-функций - myfun(x, y, S.name), myfun(x, y, C{:});
- в операциях конкатенации - [S.name], [C{:}];
- в списках выходных аргументов функции - [S.name] = myfun, [C{:}] = myfun;
- как составляющие массива ячеек - {S.name}, {C{:}}.

Примеры:

Рассмотрим некоторые примеры использования списков значений:

формирование массива ячеек:

```
C = {1 2 3 4}
C =     [1]     [2]     [3]     [4]
```

преобразование массива ячеек в числовой массив:

```
A = [C{:}]
A =     1     2     3     4
```

преобразование массива ячеек в трехмерный массив:

```
B = cat(3, C{:})
B(:, :, 1) =     1
B(:, :, 2) =     2
B(:, :, 3) =     3
B(:, :, 4) =     4
```

присвоение значений одному из полей массива записей:

```
[S(1:3).FIELD] = deal(5)
S =
3x1 struct array with fields:
    FIELD
S(:).FIELD
ans =     5
ans =     5
ans =     5
```

В результате выполненных операций были сформированы следующие массивы:

```
whos
  Name      Size          Bytes  Class
  A         1x4              32  double array
  B         1x1x4            32  double array
  C         1x4             400  cell array
  S         3x1             332  struct array
Grand total is 22 elements using 796 bytes
Общее количество элементов - 22; используют 796 байт
```

Сопутствующие функции: CAT, CELL2STRUCT, DEAL, NUM2CELL, STRUCT2CELL, VARARGIN, VARARGOUT.

VARARGIN **Список входных аргументов переменной длины**

Синтаксис:

z = function myfun(x, y, varargin)

Описание:

Переменная varargin позволяет объединить любое количество входных аргументов; она представляет собой массив ячеек, который содержит аргументы-опции вызываемой функции. Эта переменная должна быть последней в списке входов, а ее написание допускается только строчными буквами.

Пример:

Рассмотрим М-функцию

```
function myplot(x, varargin)
    plot(x, varargin{:})
```

Она объединяет все входные аргументы начиная со второго в одну переменную varargin. В свою очередь, при обращении к функции plot используется список значений varargin{:}, чтобы передать все задействованные аргументы. Например, при вызове функции myplot в форме

```
myplot(sin(0:.1:1), 'color', [.5 .7 .3], 'linestyle', ':')
```

переменная varargin - это массив ячеек размера 1×4, содержащий значения

```
'color', [.5 .7 .3], 'linestyle', ':'.
```

Сопутствующие функции: VARARGOUT, NARGIN, NARGOUT, INPUTNAME, FUNCTION, LISTS.

VARARGOUT **Список выходных аргументов переменной длины**

Синтаксис:

```
[z, varargout] = myfun(x, y, n)
```

Описание:

Переменная varargout позволяет объединить любое количество выходных аргументов; она представляет собой массив ячеек, который содержит аргументы-опции выхода функции. Эта переменная должна быть последней в списке выходов, а ее написание допускается только строчными буквами.

Переменная varargout не создается при вызове функции; ее необходимо создать при формировании выходов создаваемой М-функции, используя соответствующие операторы цикла, как это показано в нижеследующем примере.

Пример:

Рассмотрим М-функцию

```
function [s, varargout] = mysize(x)
    nout = max(nargout, 1)-1;
    s = size(x);
    for i=1:nout,
        varargout(i) = {s(i)};
    end
```

Здесь с использованием цикла объединены все выходные аргументы начиная со второго в одну переменную varargout.

Переменная varargout - это массив ячеек размера 1×nout, содержащий значения размерностей выходного многомерного массива. Например, при обращении вида

```
[s, rows, cols, pages] = mysize(rand(4, 5, 3));
```

возвращаются значения

```
s = [4 5 3], rows = 4, cols = 5, pages = 3.
```

Сопутствующие функции: VARARGIN, NARGIN, NARGOUT, FUNCTION, LISTS.

Объектно-ориентированное программирование

Введение новых типов данных и операций, определяемых пользователем, характеризует подход, известный как объектно-ориентированное программирование.

Классы и объекты позволяют добавлять новые типы данных и новые операции. *Класс* описывает тип переменной и определяет, какие операции и функции могут быть применены к переменной этого типа. *Объект* - это структура или образец некоторого класса.

В системе MATLAB определены 5 основных классов:

double - числовые массивы и матрицы, заданные в арифметике с плавающей точкой в формате удвоенной точности;

sparse - двумерные действительные или комплексные разреженные матрицы;

char - массивы символов;

struct - массивы записей (структуры);

cell - массивы ячеек.

Добавление классов осуществляется в рамках операционной среды системы MATLAB, которая обеспечивает возможность хранения созданных объектов и организации каталога М-файлов, которые определяют допустимые методы обработки для данного класса объектов. Каталог класса включает М-функции, которые определяют способ, каким операторы системы MATLAB, включая арифметические, обработки индексов, конкатенации, применяются к объекту данного класса.

Класс объектов inline

В рамках системы MATLAB 5 определен новый класс объектов INLINE. Он предназначен для описания функций в форме f(x, P1, P2, ...), соответствующей их математическому описанию. При таком представлении вычисление функции для заданных значений аргумента и параметров может быть записано в естественной форме f(3, 0.1, 0.3).

Для объектов класса INLINE определены следующие методы:

argnames	disp	formula	nargin	vectorize
cat	display	horzcat	nargout	vertcat
char	feval	inline	subsref	

INLINE/INLINE — **Создать объект класса INLINE**

Синтаксис:

g = inline('<*выражение*>')

g = inline('<*выражение*>', *'имя_1', 'имя_2'*, ...)

g = inline('<*выражение*>', n)

Описание:

Функция g = inline('<*выражение*>') формирует объект класса INLINE. Объекты этого класса - это функции, заданные в символьном виде, что позволяет обращаться к ним как привычным математическим объектам. Аргумент функции определяется автоматически путем поиска в составе выражения одноместного символа, отличного от 'i' и 'j'. Если символ отсутствует, то в качестве аргумента функции используется символ 'x'. Если в выражении присутствует несколько символов, то в качестве аргумента функции выбирается символ, ближайший в лексикографическом смысле к 'x'. Из двух символов, равноудаленных в лексикографическом смысле от 'x', выбирается в качестве аргумента тот, который в алфавите следует после 'x'.

Функция g = inline('<*выражение*>', *'имя_1', 'имя_2', ...*) формирует объект класса INLINE, аргументы которого имеют имена *'имя_1', 'имя_2', ...* . При этом в качестве имени могут применяться сочетания символов.

Функция g = inline('<*выражение*>', n) формирует объект класса INLINE, аргументы которого имеют фиксированные имена 'x', 'P1', 'P2', ..., 'PN'.

Примеры:

```
g = inline('t^2')
g =
    Inline function:
    g(t) = t^2
    g = inline('t^2+ w^2')
g =
    Inline function:
    g(w) = t^2+ w^2
g = inline('sin(2*pi*f + theta)', 'f', 'theta')
g =
    Inline function:
    g(f, theta) = sin(2*pi*f + theta)
g = inline('x^P1', 1)
g =
    Inline function:
    g(x, P1) = x^P1
```

Сопутствующие функции: CHAR.

INLINE/ARGNAMES **Аргументы объекта класса INLINE**

Синтаксис:

S = argnames(g)

Описание:

Функция S = argnames(g) возвращает аргументы функции как объекта класса INLINE в виде массива строковых ячеек.

Примеры:

```
g = inline('sin(2*pi*f + theta)', 'f', 'theta');
S = argnames(g)
S =
   'f'
   'theta'
```

Сопутствующие функции: INLINE/FORMULA.

INLINE/FORMULA **Формула объекта класса INLINE**

Синтаксис:

```
f = formula(g)
```

Описание:

Функция f = formula(g) возвращает формулу той функции, которая является объектом класса INLINE.

Примеры:

```
g = inline('sin(2*pi*f + theta)', 'f', 'theta');
f = formula(g)
f = sin(2*pi*f + theta)
```

Сопутствующие функции: INLINE/ARGNAMES, INLINE/CHAR.

INLINE/CHAR **Преобразовать объект класса INLINE в массив символов**

Синтаксис:

```
s = char(g)
```

Описание:

Функция s = char(g) возвращает формулу объекта класса INLINE в виде массива символов. Эта функция равносильна функции formula(g).

Примеры:

```
g = inline('sin(2*pi*f + theta)', 'f', 'theta');
s = char(g)
s = sin(2*pi*f + theta)
```

Сопутствующие функции: INLINE, INLINE/FORMULA.

INLINE/VECTORIZE **Преобразовать объект класса INLINE в массив символов**

Синтаксис:

```
sv = vectorize(s)
gv = vectorize(g)
```

Описание:

Функция sv = vectorize(s), где s - символьное выражение, вставляет символ . поэлементной обработки массивов перед арифметическими операциями ^, * и /, используемыми в формуле функции. Результатом является векторизованное строковое выражение.

Функция gv = vectorize(g), где g - объект класса INLINE, вставляет символ . поэлементной обработки перед арифметическими операциями ^, * и /, используемыми в формуле функции. Результатом является векторизованная формула объекта класса INLINE.

Примеры:

```
g = inline('sin(2*pi*f + theta)', 'f', 'theta');
s = char(g)
s = sin(2*pi*f + theta)
sv = vectorize(s)
sv = sin(2.*pi.*f + theta)
gv = vectorize(g)
gv =
    Inline function:
    gv(f,theta) = sin(2.*pi.*f + theta)
```

Сопутствующие функции: INLINE, INLINE/FORMULA.

Переопределение классов

Во многих случаях можно изменить поведение операторов и функций системы MATLAB, когда в качестве аргумента выступает объект. Это осуществляется путем переопределения соответствующих функций. Переопределение класса открывает возможность обработки с помощью этой функции различных типов данных при произвольном количестве входных аргументов.

Следующая таблица устанавливает символьные имена для большинства встроенных операторов системы MATLAB.

Оператор	*Имя М-файла*	*Описание*
a + b	plus(a,b)	Двоичное сложение
a − b	minus(a, b)	Двоичное вычитание
−a	uminus(a)	Унарное вычитание
+a	uplus(a)	Унарное сложение
a.*b	times(a, b)	Поэлементное умножение
a*b	mtimes(a, b)	Умножение матриц
a./b	rdivide(a, b)	Правое поэлементное деление
a.\b	ldivide(a, b)	Левое поэлементное деление
a/b	mrdivide(a, b)	Правое деление матриц
a\b	mldivide(a, b)	Левое деление матриц

a.^b	power(a, b)	Поэлементное возведение в степень
a^b	mpower(a, b)	Возведение матрицы в степень
a < b	lt(a, b)	Меньше
a > b	gt(a, b)	Больше
a <= b	le(a, b)	Меньше или равно
a >= b	ge(a, b)	Больше или равно
a ~= b	ne(a, b)	Не равно
a == b	eq(a, b)	Тождественно
a & b	and(a, b)	Логическое И
a | b	or(a, b)	Логическое ИЛИ
~a	not(a, b)	Логическое НЕ
a:d:b a:b	colon(a, d, b) colon(a, b)	Формирование вектора
a'	ctranspose(a)	Транспонирование матрицы
a.'	transpose(a)	Транспонирование массива
command window output	display(a)	Вывод на терминал
[a b]	horzcat(a, b, ...)	Объединение в строку
[a; b]	vertcat(a, b, ...)	Объединение в столбец
a(s1,s2,...sn)	subsref(a, s)	Индексная ссылка
a(s1,...,sn) = b	subsasgn(a, s, b)	Индексное выражение
b(a)	subsindex(a, b)	Индекс подмассива

Описание функций и команд

CLASS **Определить класс объекта или создать объект**

Синтаксис:

str = class('<*имя_объекта*>')
obj = class(S, '<*имя_класса*>')
obj = class(S, '<*имя_класса*>', <*родитель1*>, <*родитель2*> ...)

Описание:

Функция str = class('<*имя_объекта*>') возвращает строку, содержащую имя класса, соответствующего следующей таблице.

Имя класса	*Класс объектов*
double	Многомерные массивы чисел в арифметике с плавающей точкой в формате удвоенной точности
sparse	Двумерные действительные или комплексные разреженные матрицы
char	Массивы символов
struct	Массивы записей (структура)
cell	Массивы ячеек
'<*имя_класса*>'	Класс, определяемый пользователем

Функция obj = class(S, '<имя_класса>') создает объект класса с указанным именем, используя структуру S в качестве шаблона. Это относится только к М-функциям с именем *имя_класса*.m, размещенным в каталоге @*имя_класса*.

Функция obj = class(S, '<*имя_класса*>', <*родитель1*>, <*родитель2*> ...) создает объект класса с указанным именем, используя структуру S в качестве шаблона, а также гарантирует, что вновь создаваемый объект наследует методы и поля родительских объектов, указанных в качестве аргументов.

Сопутствующие операторы: INFERIORTO, ISA, SUPERIORTO.

ISA **Определить принадлежность объекта к данному классу**

Синтаксис:

K = isa(obj, '<*имя_класса*>')

Описание:

Функция K = isa(obj, '<*имя_класса*>') возвращает логическое TRUE (1), если объект принадлежит данному классу, и логическое FALSE (0) - в противном случае.

Аргумент '<*имя_класса*>' - это либо имя класса, определенного пользователем, либо имя одного из предопределенных классов системы MATLAB:

Имя класса	*Класс объектов*
double	Многомерные массивы чисел в арифметике с плавающей точкой в формате удвоенной точности
sparse	Двумерные действительные или комплексные разреженные матрицы
char	Массивы символов
struct	Массивы записей (структура)
cell	Массивы ячеек

Пример:

Функция

isa(rand(3, 4), 'double')

истинна и возвращает значение

ans = 1

Сопутствующие операторы: CLASS.

ISOBJECT **Выявление объекта некоторого класса**

Синтаксис:

k = isobject(A)

Описание:

Функция k = isobject(A) возвращает логическое TRUE (1), если A - объект некоторого класса, и логическое FALSE (0) - в противном случае.

Сопутствующие операторы: операторы группы IS*.

METHODS **Вывести на терминал список методов для данного класса**

Синтаксис:

methods <*имя_класса*>

s = methods('<*имя_класса*>')

Описание:

Команда methods <*имя_класса*> выводит на терминал список методов для данного класса.

Функция s = methods('<*имя_класса*>') возвращает массив ячеек, содержащих имена методов в виде строковых величин.

Пример:

Команда methods polynom выводит на экран следующий список методов для класса polynom:

```
methods polynom
Methods for class polynom:
Методы для класса  polynom:
char    display minus   plot    polynom roots
diff    double  mtimes  plus    polyval
```

Функция s = methods('polynom') выводит на экран следующий список методов для класса polynom в виде массива ячеек символов:

```
s = methods( 'polynom')
s =
    'polynom'
    'mtimes'
    'plus'
    'plot'
    'display'
    'diff'
    'char'
    'roots'
    'minus'
    'double'
    'polyval'
```

Сопутствующие операторы: HELP, WHAT, WHICH.

INFERIORTO **Отношение низшего класса**

Синтаксис:

inferiorto('<*имя_класса1*>', '<*имя_класса2*>', ...)

Описание:

Команда inferiorto('<*имя_класса1*>', '<*имя_класса2*>', ...), вызванная внутри конструктора класса (например, myclass.m), определяет, что метод myclass.m не должен вызываться, если функция вызвана с объектом класса myclass и с одним или более объектами классов '<*имя_класса1*>', '<*имя_класса2*>', ...

Пояснение:

Допустим, что A - это объект класса 'class_a', B - это объект класса 'class_b' и C - это объект класса 'class_c'. Допустим также, что конструктор class_c.m содержит утверждение inferiorto('class_a'), что означает низший приоритет класса 'class_c' по отношению к классу 'class_a'. Тогда при вызове функций e = fun(a, c) или e = fun(c, a) вызывается функция class_a/fun.

Если функция вызывает два объекта, отношение классов которых не определено, то оба объекта имеют одинаковый приоритет и используется метод, относящийся к первому объекту в списке аргументов вызываемой функции. То есть вызов fun(b, c) использует метод class_b/fun, а вызов fun(c, b) использует метод class_c/fun.

Сопутствующие операторы: SUPERIORTO.

SUPERIORTO **Отношение высшего класса**

Синтаксис:

superiorto('*<имя_класса1>*', '*<имя_класса2>*', ...)

Описание:

Команда superiorto('*<имя_класса1>*', '*<имя_класса2>*', ...), вызванная внутри конструктора класса (например, myclass.m), определяет, что метод myclass.m должен быть вызван, если функция вызвана с объектом класса myclass и с одним или более объектами классов '*<имя_класса1>*', '*<имя_класса2>*', ...

Пояснение:

Допустим, что A - это объект класса 'class_a', B - это объект класса 'class_b' и C - это объект класса 'class_c'. Допустим также, что конструктор class_c.m содержит утверждение superiorto('class_a'), что означает высший приоритет класса 'class_c' по отношению к классу 'class_a'. Тогда при вызове функций e = fun(a, c) или e = fun(c, a) вызывается функция class_c/fun.

Если функция вызывает два объекта, отношение классов которых не определено, то оба объекта имеют одинаковый приоритет и используется метод, относящийся к первому объекту в списке аргументов вызываемой функции. То есть вызов fun(b, c) использует метод class_b/fun, а вызов fun(c, b) - метод class_c/fun.

Сопутствующие операторы: INFERIORTO.

HORZCAT, VERTCAT **Объединение объектов**

Синтаксис:

C = horzcat(A, B) C = vertcat(A, B)
Y = horzcat(X1, X2, X3, ...) Y = vertcat(X1, X2, X3, ...)

Описание:

Функция C = horzcat(A, B) реализует горизонтальное объединение двух объектов [A B].

Функция Y = horzcat(X1, X2, X3, ...) реализует горизонтальное объединение конечного числа объектов [X1 X2 X3 ...].

Функция C = vertcat(A, B) реализует вертикальное объединение двух объектов [A; B].

Функция Y = vertcat(X1, X2, X3, ...) реализует вертикальное объединение конечного числа объектов [X1; X2; X3; ...].

Сопутствующие операторы: CAT.

Переопределяемые методы:

help inline/horzcat.m.

help inline/vertcat.m

SUBSREF **Индексная ссылка**

Синтаксис:

B = subsref(A,S)

Описание:

Использование индексов или указателей полей, когда объект применяется в правой части оператора присваивания, называется *индексной ссылкой*. Примерами индексных ссылок являются выражения вида A(i), A{i}, A.<*поле*>.

Функция B = subsref(A, S) вызывается при обработке выражений A(i), A{i}, A.<*поле*>, где A - объект, а S - массив записей со следующими полями:

Поле	*Назначение*
type	Строка, содержащая группы символов ( ), { }, ., которые определяют тип индексов: круглые скобки ( ) соответствуют числовому массиву; фигурные скобки { } - массиву ячеек, точка . - массиву записей
subs	Массив ячеек или строка, содержащая реально используемые символы, при этом разделитель : передается как строка

Пример:

Выражение A(1:2, :) вызывает метод subsref(A, S), где S - массив записей размера 1×1 с полями

S.type = ' () '

S.subs = {1:2, ':'}.

Выражение A{1:2} вызывает метод subsref(A, S), где S - массив записей размера 1×1 с полями

S.type = ' {} '

S.subs = {1:2}.

Выражение A.<поле> вызывает метод subsref(A, S), где S - массив записей размера 1×1 с полями

S.type = ' . '

S.subs = ' <*поле*> '.

Эти простые случаи являются основой для формирования более сложных выражений.

Выражение A(1, 2).name(3:4) вызывает метод subsref(A, S), где S - массив записей размера 3×1 со следующими полями:

S(1).type = ' () '	S(2).type = ' . '	S(3).type = ' () '
S(1).subs = ' {1,2} '	S(2).subs = ' name '	S(3).subs = ' {3:4} '

Сопутствующие функции: SUBSASGN, SUBSTRUCT, PAREN, SUBSINDEX, LISTS.

Переопределяемые методы:

help inline/subsref.m
help scribehgobj/subsref.m
help scribehandle/subsref.m
help hgbin/subsref.m
help figobj/subsref.m
help fighandle/subsref.m
help axistext/subsref.m
help axisobj/subsref.m
help activex/subsref.m

SUBSASGN **Индексное присваивание**

Синтаксис:

A = subsasgn(A, S, B)

Описание:

Использование индексов или указателей полей, когда объект применяется в левой части оператора присваивания, называется *индексным присваиванием.* Примерами таких выражений являются выражения вида A(i) = B, A{i} = B, A.<*поле*> = B.

Функция A = subsasgn(A, S, B) вызывается при обработке выражений A(i) = B, A{i} = B, A.<*поле*> = B, где S - массив записей со следующими полями:

Поле	*Назначение*
type	Строка, содержащая группы символов ( ), { }, ., которые определяют тип индексов: круглые скобки ( ) соответствуют числовому массиву; фигурные скобки { } - массиву ячеек, точка . - массиву записей
subs	Массив ячеек или строка, содержащая реально используемые символы, при этом разделитель : передается как строка

Пример:

Выражение A(1:2, :) = B вызывает метод A = subsasgn(A, S, B), где S - массив записей размера 1×1 с полями

S.type = ' () '
S.subs = {1:2, ':'}.

Выражение A{1:2} = B вызывает метод A = subsasgn(A, S, B), где S - массив записей размера 1×1 с полями

S.type = '{}'

S.subs = {1:2}.

Выражение A.<*поле*> = B вызывает метод A = subsasgn(A, S, B), где S - массив записей размера 1×1 с полями

Эти простые случаи являются основой для формирования более сложных выражений.

Выражение A(1, 2).name(3:4) = B вызывает метод A = subsasgn(A, S, B), где S - массив записей размера 3×1 со следующими полями:

S(1).type = '()'	S(2).type = '.'	S(3).type = '()'
S(1).subs = '{1,2}'	S(2).subs = 'name'	S(3).subs = '{3:4}'

Сопутствующие функции: SUBSREF, SUBSTRUCT, PAREN, SUBSINDEX, LISTS.

Переопределяемые методы:

help scribehgobj/subsasgn.m
help scribehandle/subsasgn.m
help hgbin/subsasgn.m
help figobj/subsasgn.m
help fighandle/subsasgn.m
help axistext/subsasgn.m
help axisobj/subsasgn.m
help activex/subsasgn.m

SUBSINDEX

Индексный дескриптор

Синтаксис:

I = subsindex(A)

Описание:

Функция I = subsindex(A) вызывается в случае выражений вида X(A), где A - объект, а X - один из встроенных типов (чаще всего double). Она должна возвратить значение объекта как целочисленный индекс в диапазоне от 0 до prod(size(X))-1). Эта функция вызывается по умолчанию функциями subsref и subsasgn.

Функция subsindex вызывается отдельно для каждого элемента списка аргументов в выражениях вида X(A, B).

Примечание. В версиях системы MATLAB 5.0-5.2 эта функция подавлена.

Сопутствующие функции: SUBSREF, SUBSASGN.

4. ОПЕРАТОРЫ, КОНСТАНТЫ, СЛУЖЕБНЫЕ СИМВОЛЫ И ПЕРЕМЕННЫЕ

Язык MATLAB - это язык операторов. Операторы, вводимые пользователем в командной строке, исполняются системой MATLAB в режиме интерпретации. Операторы имеют две формы записи:

с неявным присваиванием (команда)	с явным присваиванием (функция)
выражение	*переменная = выражение*

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.

Для изменения стандартного порядка выполнения операций используют круглые скобки, как при обычных математических выкладках. Пробелы вокруг символов операций необязательны, но их часто включают для наглядности.

Результат вычисления выражения присваивается заданной переменной в левой части от знака равенства для дальнейшего использования и, если после выражения не стоит точка с запятой, выводится на экран.

При записи нескольких операторов в одной строке следует использовать разделители. После единственного или последнего оператора строки разделитель можно не ставить.

При использовании операторов с неявным присваиванием система MATLAB автоматически создает переменную с именем ans (ANSwer) и присваивает ей значение результата. Переменная ans сохраняет значение результата выполнения только последнего оператора с неявным присваиванием.

Длина буфера командной строки ограничена 256 символами. Если оператор настолько сложен, что полностью не помещается на одной строке или по каким-либо причинам его оставшуюся часть требуется перенести на новую строку, следует ввести несколько точек (не менее двух), нажать клавишу Enter и продолжить ввод оператора в следующей строке.

Имена переменных и функций могут быть составлены из любых символов алфавита системы MATLAB, кроме специальных. Имя обязательно должно начинаться с буквы, за которой может следовать произвольное сочетание букв и цифр.

Система MATLAB позволяет работать с константами, переменными, массивами, содержащими как числовую, так и символьную информацию.

Для записи действительных чисел используется как обычная десятичная форма с необязательными десятичной точкой для целых чисел и плюсом для положительного числа, так и показательная форма с мантиссой и показателем степени по основанию 10 (отделяется от мантиссы символом e или E) без пробелов между ними. Наличие пробела перед экспоненциальной частью числа приводит к ошибке.

Вычисления в системе MATLAB ведутся как в поле вещественных, так и в поле комплексных чисел с удвоенной точностью.

Арифметические и логические операторы

+ - * / \ ^ '

Арифметические операторы. Операции над массивами, матрицами и объектами

Синтаксис:

Операции над массивами		*Операции над матрицами*	
+ A	uplus(A)	+ A	uplus(A)
- A	uminus(A)	- A	uminus(A)
A + B	plus(A, B)	A + B	plus(A, B)
A - B	minus(A, B)	A - B	minus(A, B)
A .* B	times(A, B)	A * B	mtimes(A, B)
A ./ B	rdivide(A, B)	A / B	mrdivide(A, B)
A .\ B	ldivide(A, B)	A \ B	mldivide(A, B)
A .^ B	power(A, B)	A ^ B	mpower(A, B)
A.'	transpose(A)	A'	ctranspose(A, B)

Описание:

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

	Унарный плюс: +A **uplus(A)**	Преобразование массива в самое себя; для объекта - создание объекта +A
	Унарный минус: -A **uminus(A)**	Изменение знака элементов массива; для объекта - создание объекта -A
+	**Сложение: A + B** **plus(A, B)**	Слагаемые должны быть одинакового размера за исключением случая, когда одно из них скаляр; скаляр добавляется ко всем элементам другого операнда; для объектов - создание объекта A+B
-	**Вычитание: A - B** **minus(A, B)**	Уменьшаемое и вычитаемое должны быть одинакового размера за исключением случая, когда одно из них скаляр; скаляр вычитается из всех элементов другого операнда; для объектов - создание объекта A-B
*	**Умножение матриц:** **A * B** **times(A, B)**	При умножении матриц или матриц и векторов число столбцов первого сомножителя должно быть равно числу строк второго. На скаляр умножаются все элементы сомножителя

.*	**Умножение массивов: A .* B mtimes(A, B)**	Поэлементное перемножение двух массивов одинакового размера. На скаляр умножаются все элементы массива
\	**Решение систем линейных уравнений: AX = B mldivide(A, B)**	Если **A** - квадратная матрица размера n×n и **B** - матрица размера n×k, уравнение решается методом исключения Гаусса. Если **A** - прямоугольная матрица размера m×n и **B** - матрица размера n×k, то система оказывается недоопределенной или переопределенной и решается на основе минимизации второй нормы невязок. Ранг r = rank(A) вычисляется, используя QR-разложение с выбором ведущего элемента. Такое решение будет иметь самое большее r ненулевых элементов на столбец. Если r < n, то найденное решение, как правило, не будет совпадать с pinv(A)*B, которое имеет наименьшую норму невязок. (См. описание функции pinv)
.\	**Левое деление массивов A .\ B ldivide(A, B)**	Результатом является массив с элементами B(i, j)/A(i, j); массивы должны быть одинаковых размеров за исключением случая, когда один из них вырождается в скаляр
/	**Решение систем линейных уравнений: XA = B mrdivide(A, B)**	Операция B/A равносильна операции B*inv(A), если A имеет полный ранг; если нет, то B/A = (A`\B`)`. (См. замечания для операции \)
./	**Правое деление массивов: A ./ B rdivide(A, B)**	Результатом является массив с элементами A(i, j)/B(i, j); массивы должны быть одинаковых размеров за исключением случая, когда один из них вырождается в скаляр
^	**Степень матрицы: A^p mpower(A, B)**	Если p - целое положительное число, то степень матрицы вычисляется путем перемножения ее на себя; если p - целое отрицательное число, то то же самое относится к обратной матрице. Для других значений p вычисляются собственные значения и векторы, так что если [R, D] = eig(A), то A^p = R*D.^p/R
.^	**Степень массива: A .^ B power(A, B)**	Результатом является массив с элементами A(i, j)^B(i, j); массивы должны быть одинаковых размеров за исключением случая, когда один из них вырождается в скаляр
'	**Транспонирование матрицы: A' ctranspose(A)**	Для действительных - результатом является транспонированная матрица; для комплексных - транспонирование дополняется комплексным сопряжением
.'	**Транспонирование массива: A .' transpose(A)**	Для действительных и комплексных массивов строки просто заменяются столбцами; операция комплексного сопряжения не выполняется

Пример:

В качестве примера рассмотрим два одномерных массива (вектора), над которыми совершаются описанные выше операции; для печати использован формат format rat.

Операции над матрицами				*Операции над массивами*			
x	1			y	4		
	2				5		
	3				6		
x'	1	2	3	y'	4	5	6
x + y 5				x - y	-3		
	7				-3		
	9				-3		
x + 2 3				x - 2	-1		
	4				0		
	5				1		
x * y	Error			x .* y 4			
					10		
					18		
x' * y	32			x' .* y		Error	
x * y	4	5	6	x .* y'		Error	
	8	10	12				
	12	15	18				
x * 2 2				x .* 2	2		
	4				4		
	6				6		
x \ y	16/7			x .\ y 4			
					5/2		
					2		
2 \ x	1/2			2 .\ x 2			
	1				1		
	3/2				2/3		
x / y	0	0	1/6	x ./ y 1/4			
	0	0	1/3		2/5		
	0	0	1/2		1/2		
x / 2	1/2			x ./ 2 1/2			
	1				1		
	3/2				3/2		
x ^ y	Error			x .^ y 1			
					32		
					729		
x ^ 2	Error			x .^ 2 1			
					4		
					9		

2 ^ x	Error	2 .^ x2	4 8	
(x + i*y)'		1 – 4i	2 – 5i	3 – 6i
(x + i*y).'		1 + 4i	2 + 5i	3 + 6i

<	≤	>	≥	==	~=
LT	**LE**	**GT**	**GE**	**EQ**	**NE**

Логические операторы. Операции отношения

Синтаксис:

A < B	lt(A, B)
A <= B	le(A, B)
A > B	gt(A, B)
A >= B	ge(A, B)
A == B	eq(A, B)
A ~= B	ne(A, B)

Описание:

Операции отношения выполняют поэлементное сравнение двух массивов; они возвращают в качестве результата массив того же размера, элементы которого равны единице, если эти элементы совпадают, и нулю, если они не совпадают.

Операции <, ≤, >, ≥ используют для сравнения только действительные части комплексных элементов; операции ==, ~= осуществляют сравнение как действительных, так и мнимых частей.

Для сравнения двух строк следует пользоваться функцией strcmp.

Функциональная форма операций отношения применяется в рамках объектно-ориентированного программирования.

Пример:

Если один из операндов логического оператора скаляр, а не массив, то скалярный операнд расширяется до размеров этого массива. Следующие две последовательности операторов равносильны:

```
                X = 5;     X >= [1 2 3; 4 5 6; 7 8 10]
X = 5*ones(3,3);    X >= [1 2 3; 4 5 6; 7 8 10]
```

Результатом является следующий массив:

```
X =
   1   1   1
   1   1   0
   0   0   0
```

Сопутствующие функции и операторы: FIND, ANY, ALL, STRCMP, &, |, ~.

& | ~
AND OR NOT
XOR

Логические операции

Синтаксис:

A & B	and(A, B)
A | B	or(A, B)
~A	not(A)
	xor(A, B)

Описание:

Для логических операций устанавливаются следующие соответствия:

Логическая операция	*Символ операции*	*Обозначение в MATLAB*	
		Оператор	*Функция*
ЛОГИЧЕСКОЕ НЕ (отрицание)	~A	~A	not(A)
ЛОГИЧЕСКОЕ И (конъюнкция)	A⊗B	A & B	and(A, B)
НЕИСКЛЮЧАЮЩЕЕ ИЛИ (дизъюнкция)	A+B	A | B	or(A, B)
ИСКЛЮЧАЮЩЕЕ ИЛИ (сложение по модулю 2)	A⊕B	-	xor(A, B)

Таблица истинности для этих операторов и функций имеет следующий вид:

Входы		*not(A)*	*and(A, B)*	*or(A, B)*	*xor(A, B)*
A	B	~A	A & B	A | B	
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

При обработке массивов эти операции применяются поэлементно. В случае числового массива значению 0 соответствует булево значение FALSE, а любому другому значению - булево значение TRUE.

Все логические операции могут быть использованы при обработке объектов, если применяется функциональная форма оператора.

Логические операции имеют низший приоритет по отношению к операциям отношения и арифметическим операциям. Однако среди логических операций наивысший приоритет имеет операция NOT; операции AND и OR имеют одинаковый приоритет и выполняются слева направо.

Сопутствующие функции и операторы: FIND, ANY, ALL,
<, <=, >, >=, ==, ~=.

Специальные символы, переменные и константы

[ ] ( ) = ' .
, ; : % !

Специальные символы

Синтаксис:

[ ] ( ) = ' . , ; : % !

Описание:

[ ]	**Квадратные скобки**	Формирование векторов и матриц. Указание последовательности выходных параметров при обращении к функциям, возвращающим более одного параметра
( )	**Круглые скобки**	Указание порядка выполнения операций в арифметических выражениях. Указание индексов элемента вектора, матрицы, массива. Указание последовательности входных параметров функции
=	**Знак присваивания**	Знак присваивания в арифметических выражениях
'	**Транспонирование матриц**	Транспонирование, сопровождаемое комплексным сопряжением для комплексных матриц
.	**Разделитель - *точка***	Выполняет несколько функций: а) десятичная точка; б) поэлементное выполнение операций над массивом данных
..	" "	Переход по дереву каталогов на один уровень вверх
... и более	" "	Признак продолжения строки
, [,]	**Разделитель - *запятая***	Используется: а) для отделения операторов языка MATLAB; б) для указания индексов элемента вектора, матрицы, массива
; [;]	**Разделитель - *точка с запятой***	Используется: а) для подавления вывода на экран результата вычислений; б) для отделения строк матриц или массивов
: [:]	**Разделитель - *двоеточие***	Используется: а) для формирования векторов, выделения строк, столбцов, подблоков массива; б) в заголовке цикла for

%	**Указатель**	Является указателем: а) логического конца строки; следующий за ним текст игнорируется; б) строки комментария
!	**Указатель**	Является указателем ввода команды DOS

Сопутствующие функции: арифметические и логические операторы.

ANS **Результат выполнения последней операции**

Описание:

При использовании операторов с неявным присваиванием система MATLAB автоматически создает переменную ans(ANSwer) и присваивает ей значение результата. Переменная ans сохраняет свое значение до следующего оператора с неявным присваиванием.

Пример:

После ввода выражения

sin(pi/6)

система MATLAB выдаст результат

ans = 0.50000000000000

I, J **Мнимая единица**

Описание:

Константам I и J первоначально присваивается значение, равное sqrt(-1). Они используются для ввода комплексных чисел.

Пример:

При вводе комплексных чисел допустимы следующие формы записи:

3+2i, 3+2*i, 3+2i, 3+2*j или 3+2*sqrt(-1)

Приведенные выражения представляют одно и то же комплексное число.

Следует отметить, что с символами I и J могут также быть ассоциированы и другие величины - например, переменная цикла for или индекс элемента массива.

INF **Бесконечность**

Синтаксис:

Inf

Описание:

Inf - специальная переменная, позволяющая фиксировать переполнение разрядной сетки ПЭВМ. Этот механизм очень редко предоставляется пользователю для работы, однако в системе MATLAB это обеспечено благодаря возможностям арифметического сопроцессора.

Пример:

При выполнении операции деления на нуль

1.0/0.0

машина выдаст сообщение

Warning: Divide by zero

Предупреждение: Деление на нуль

ans = Inf

Аналогичный результат будет получен при выполнении следующей операции, когда происходит переполнение разрядной сетки:

exp(1000)

ans = Inf

Сопутствующие переменные и функции: NaN, ISINF, ISFINITE.

NaN **Нечисловое значение**

Синтаксис:

NaN

Описание:

NaN - специальная переменная, принятая в стандарте IEEE-арифметики, которая позволяет фиксировать результат, непредставимый в виде числа. Следующие операции имеют результат в виде переменной NaN:

- умножение вида 0*Inf;
- деление вида 0/0 и Inf/Inf;
- сложение и вычитание бесконечностей, например: (+Inf) – (–Inf);
- вычисление функции остатка rem(x, y) для y = 0 или x = Inf;
- вычисление отношений с предикатами < или >, например функций вида min([Inf NaN]) и max([Inf NaN]);
- любые арифметические операции с переменной NaN.

Сопутствующие переменные и функции: INF, ISNAN.

PI **Число π (3.1415926535897....)**

Описание:

Константа, рассчитываемая по одной из следующих формул:

PI = 4*atan(1) = imag(log(-1)) = 3.1415926535897....

Характеристики арифметики с плавающей точкой

EPS **Машинная точность**

Синтаксис:

eps

Описание:

Переменная eps определяет относительную точность операций с плавающей точкой, которая часто используется для оценки сходимости итерационных методов. Машинная точность eps - это наименьшее число, для которого на данной ЭВМ еще выполняется соотношение 1.0 + eps > 1.0. Для арифметического устройства, соответствующего стандарту IEEE, ее значение равно eps = 2^{-52}, что приближенно составляет 2.220446e-016. В процессе работы в системе MATLAB пользователь может присвоить этой специальной переменной любое значение, включая 0.

Сопутствующие переменные и функции: REALMIN, REALMAX, ISIEEE.

LOG2 **Представление числа в формате с плавающей точкой**

Синтаксис:

[f, e] = log2(x)

Описание:

Функция [f, e] = log2(x) представляет действительное число x в нормализованной форме с основанием степени 2:

x = f * 2^e , 0.5 ≤ abs(f) ≤ 1 ;

0 = 0 * 2^0 .

В случае, когда X - массив, справедливо представление

X = f .* 2.^e .

Пример:

Основные константы языка MATLAB имеют следующее представление в IEEE-арифметике:

x	*f*	*e*
1	1/2	1
-1	-1/2	1
pi	pi/4	2
eps	1/2	-51
realmax	1-eps/2	1024
realmin	1/2	-1021

Сопутствующие функции: LOG, LOG10, POW2, NEXTPOW2, REALMAX, REALMIN .

POW2 **Преобразование нормализованного числа в стандартный формат**

Синтаксис:

x = pow2(f, e)

Описание:

Функция x = pow2(f, e) формирует действительное число x по его мантиссе f и показателю степени e исходя из его нормализованного представления по основанию 2:

x = f * 2^e.

Пример:

Нормализованные представления основных констант языка MATLAB и их стандартный формат в IEEE-арифметике:

f	*e*	*x*
1/2	1	1
-1/2	1	-1
pi/4	2	pi
1/2	-51	eps
1-eps/2	1024	realmax
	-1021	realmin

Сопутствующие функции: LOG2, REALMIN, REALMAX.

REALMAX, REALMIN **Наименьшее и наибольшее положительные числа с плавающей точкой**

Синтаксис:

huge = realmax
tiny = realmin

Описание:

Константа realmax - наибольшее число в формате с плавающей точкой для данного компьютера; любое большее значение соответствует переменной Inf.

Константа realmin - наименьшее нормализованное положительное число в формате с плавающей точкой для данного компьютера, принятое в системе MATLAB.

Алгоритм:

Значения realmax и realmin вычисляются с помощью функции pow2 согласно следующим соотношениям:

realmax = pow2(2 - eps, maxexp);
realmin = pow2(1, minexp),

где значения maxexp и minexp зависят от типа компьютера.

Для компьютеров с IEEE-арифметикой:

maxexp = 1023, realmax = 1.797693134862316e+308;
minexp = -1022, realmin = 2.225073858507202e-308.

Сопутствующие функции: EPS.

FLOPS **Подсчет количества выполненных операций с плавающей точкой**

Синтаксис:

f = flops
flops(0)

Описание:

Фуикция f = flops возвращает общее число выполненных операций с плавающей точкой.

Команда flops(0) сбрасывает счетчик операций в нуль.

Алгоритм:

В интерпретаторе системы MATLAB приняты следующие правила подсчета числа операций с плавающей точкой: сложение или вычитание двух действительных чисел - 1 flop, комплексных - 2 flops; умножение или деление - 1 flop, если результат действительный, и 6 flops, если результат комплексный. Для вычисления действительного значения элементарной функции требуется 1 flop.

При этом следует иметь в виду, что учитываются не все, а только наиболее значимые операции с плавающей точкой.

Если A и B - действительные матрицы размера n×n, то указанные ниже операции имеют следующую производительность:

Операция	*Производительность, flops*
A + B	n^2
A * B	2*n^3
A ^ 100	99*2*n^3
LU(A)	(2/3)*n^3

Пример:

Для оценки производительности системы MATLAB при решении задач линейной алгебры может быть использована следующая программа:

```
n = 100; A = rand(n, n); b = rand(n, 1);
flops(0)
tic; x = A\b; t = toc
megaflops = flops/t/1.e6
```

Функции вычисления времени и дат

Система MATLAB включает следующие функции для обработки времени и дат:

Тип функции	*Имя функции*	*Назначение*
Время и дата	Now	Текущее время и дата в форме числа
	Date	Текущая дата в форме строки
	Clock	Текущее время и дата в форме вектора
Преобразование	Datenum	Перевести в номер порядковой даты
	Datestr	Строковое представление даты
	Datevec	Векторное представление даты
День и дата	Calendar	Календарь
	Weekday	День недели
	Eomday	Последний день месяца
	Datetick	Дата с метками времени
Интервалы времени	Cputime	Время работы центрального процессора
	Tic	Начало отсчета
	Toc	Конец отсчета
	Etime	Прошедшее время

Эти функции размещены в каталоге timefun.

Форматы дат

Система MATLAB работает с тремя форматами даты: *строковым, числовым* и *векторным*.

Внутреннее представление даты числовое и соответствует количеству дней, прошедших с некоторой фиксированной даты, в качестве которой в системе MATLAB принять 1 января 0000 года. Числовой формат даты отсчитывается от полуночи, то есть 18 часов соответствует 0.75 дня. Таким образом, дате '10-Nov-1997, 6:00 pm' в строковом формате соответствует числовой формат 729284.75.

Все функции, использующие дату, работают как со строковым, так и с числовым форматом. При работе в командной строке предпочтителен строковый формат; если же необходимо выполнять вычисления, то эффективнее числовой формат.

Для некоторых функций системы MATLAB внутренним форматом даты является *векторный* формат, который состоит из следующих элементов [*год месяц день час минута секунда*].

NOW — **Текущее время и дата в форме числа**

Синтаксис:

```
t = now
```

Описание:

Функция t = now выводит текущее время и дату в форме числа.

Функция floor(now) позволяет выделить из числового формата текущую дату, а функция rem(now, 1) - текущее время.

Для представления текущего времени и даты в форме строки следует использовать функцию datestr(now).

Пример:

```
now
ans = 729952.368728588
floor( now)
ans = 729952
rem(now,1)
ans = 0.368738773162477
datestr(now)
ans = 16-Jul-1998 08:50:59
```

Сопутствующие функции: DATE, DATENUM, DATESTR, CLOCK.

DATE — **Дата**

Синтаксис:

```
s = date
```

Описание:

Функция s = date возвращает символьную строку, содержащую календарную дату в формате [*день - месяц - год*].

Пример:

```
s = date
s = 16-Jul-1998
```

Сопутствующие функции: NOW, CLOCK, DATENUM.

CLOCK **Часы**

Синтаксис:

```
c = clock
```

Описание:

Функция c = clock возвращает вектор-строку из шести элементов, определяющих текущее время и дату в десятичном формате

[*год месяц день час минута секунда*].

Первые 5 элементов являются целыми числами, шестой - имеет несколько десятичных знаков после точки в зависимости от установленного командой format формата вывода.

Функция fix(clock) выдает целочисленные показания часов, округляя значение секунд до целых.

Пример:

```
format short g
clock
ans = 1998      7      16      9      12      8.8
fix(clock)
ans = 1998      7      16      9      12      9
```

Это соответствует 9:12:09 16.07.1998

Сопутствующие функции: DATEVEC, DATENUM, NOW, ETIME, TIC, TOC, CPUTIME.

Преобразование форматов дат

Для преобразования форматов дат предназначены следующие функции:

datestr	Строковый формат даты
datenum	Числовой формат даты
datevec	Векторный формат даты

DATESTR **Строковый формат даты**

Синтаксис:

s = datestr(D, <*номер_формата*>)

Описание:

Функция s = datestr(D, <*номер_формата*>) преобразовывает числовой формат даты D в один из 19 выходных строковых форматов даты и времени, указанных в таблице.

По умолчанию для строкового представления даты используются форматы 0, 16 или 1 в зависимости от того, содержит ли числовой формат только дату, только время или то и другое вместе.

Номер формата	*Формат*	*Описание*
0	01-Mar-1996	День-месяц-год
1	01-Mar-1996 15:45:17	День-месяц-год час:минута:секунда
2	03/01/96	Месяц/день/год
3	Mar	Месяц, (3 буквы)
4	M	Месяц, (1 буква)
5	3	Месяц
6	03/01	Месяц/день
7	1	День месяца
8	Wed	День недели, (3 буквы)
9	W	День недели, (1 буква)
10	1996	Год, (4 цифры)
11	96	Год, (2 цифры)
12	Mar96	Месяц год
13	15:45:17	Час: минута:секунда
14	03:45:17 PM	Час:минута:секунда (AM или PM)
15	15:45	Час:минута
16	03:45 PM	Час:минута (AM или PM)
17	Q1-96	Квартал - год
18	Q1	Квартал

Пример:

```
datestr(floor(now))
ans = 16-Jul-1998
datestr(rem(now, 1))
ans = 9:40 AM
datestr(now)
ans = 16-Jul-1998 09:40:26
```

Сопутствующие функции: DATE, DATENUM, DATEVEC.

DATENUM **Числовой формат даты**

Синтаксис:

N = datenum(s)
N = datenum(<*год*>, <*месяц*>, <*день*>)
N = datenum(<*год*>, <*месяц*>, <*день*>, <*час*>, <*мин.*>, <*сек.*>)

Описание:

Функция N = datenum(s) преобразовывает строковое представление даты в числовое, причем значение даты 1 соответствует **1 января 0000 года**. Строка s должна быть представлена в одном из форматов 0, 1, 2, 6, 13, 14, 15, 16 функции datestr. Строки дат, содержащие только два символа года, считаются принадлежащими текущему столетию.

Функция N = datenum(<*год*>, <*месяц*>, <*день*>) возвращает порядковый номер даты, когда аргументы являются строками одинаковой длины либо скалярами.

Функция N = datenum(<*год*>, <*месяц*>, <*день*>, <*час*>, <*мин.*>, <*сек.*>) возвращает десятичное число с мантиссой, когда аргументы являются строками одинаковой длины либо скалярами. Если значение одного из аргументов вышло за допустимый диапазон, осуществляется необходимое преобразование, то есть секунды преобразовываются в минуты, минуты - в часы и т. д.

Пример:

```
n = datenum(datestr(now))
n = 729952.430613426
n = datenum(datestr(floor(now)))
n = 729952
datestr(datenum(2001, 2, 29))
ans = 01-Mar-2001
datestr(datenum(2000, 2, 29))
ans = 29-Feb-2000
```

Здесь в системе MATLAB допущена ошибка, поскольку год 2000 не является високосным.

Сопутствующие функции: NOW, DATESTR, DATEVEC.

DATEVEC **Векторный формат даты**

Синтаксис:

c = datevec(T)
[<*год*>, <*месяц*>, <*день*>, <*час*>, <*мин.*>, <*сек.*>] = datevec(T)

Описание:

Функция c = datevec(T) преобразовывает строковое и числовое представление даты в векторное, когда вектор c включает следующие компоненты: [<*год*>, <*месяц*>, <*день*>, <*час*>, <*мин.*>, <*сек.*>].

Если переменная T является строкой, то она должна быть представлена в одном из форматов 0, 1, 2, 6, 13, 14, 15, 16 функции datestr. Строки дат, содержащие только два символа года, считаются принадлежащими текущему столетию.

Функция [<*год*>, <*месяц*>, <*день*>, <*час*>, <*мин.*>, <*сек.*>] = datevec(T) возвращает результат в виде отдельных компонентов.

Пример:

```
datevec(datestr(now))
ans = 1998      7      16      11      2      52
datevec(datestr(floor(now)))
ans = 1998      7      16      0      0      0
datevec(datestr(rem(now, 1)))
ans =   0   0   0   11   3   0
datevec(now)
ans = 1998      7      16      11      10      25.83
datevec(floor(now))
ans = 1998      7      16      0      0      0
```

Сопутствующие функции: DATENUM, DATESTR, CLOCK.

День и дата

Для представление дней и дат предназначены следующие функции:

Calendar	Календарь
Weekday	День недели
Eomday	Последний день месяца
Datetick	Разметка графических осей по формату даты

CALENDAR **Календарь**

Синтаксис:

```
c = calendar
c = calendar(d)
c = calendar(y, m)
calendar(...)
```

Описание:

Функция c = calendar возвращает массив размера 6×7, который содержит календарь текущего месяца, начинающийся с воскресенья (первый столбец) и заканчивающийся субботой.

Функция c = calendar(<*дата*>), где дата может быть задана в строковом или числовом формате, возвращает календарь соответствующего месяца.

Функция c = calendar(<*год*>, <*месяц*>), где год и месяц заданы в виде целых чисел, возвращает календарь указанного месяца и года.

Команда calendar(...) выводит календарь на экран терминала.

Пример:

```
calendar(now)
              Jul 1998
   S   M   Tu   W   Th   F   S
   0   0   0    1   2    3   4
   5   6   7    8   9    10  11
   12  13  14   15  16   17  18
   19  20  21   22  23   24  25
   26  27  28   29  30   31  0
   0   0   0    0   0    0   0
calendar(2000, 2)
              Feb 2000
   S   M   Tu   W   Th   F   S
   0   0   1    2   3    4   5
   6   7   8    9   10   11  12
   13  14  15   16  17   18  19
   20  21  22   23  24   25  26
   27  28  29   0   0    0   0
   0   0   0    0   0    0   0
```

Сопутствующие функции: DATENUM.

WEEKDAY **День недели**

Синтаксис:

[<*номер_дня*>, <*имя_дня*>] = weekday(T)

Описание:

Функция [<*номер_дня*>, <*имя_дня*>] = weekday(T) возвращает <*номер_дня*> в числовом, а <*имя_дня*> - в строковом формате согласно следующей таблице:

<*номер_дня*>	<*имя_дня*>
1	Sun
2	Mon
3	Tue
4	Wed
5	Thu
6	Fri
7	Sat

Пример:

```
[d, w] = weekday(now)
d =   5
w = Thu
```

Сопутствующие функции: EOMDAY, DATENUM, DATEVEC.

EOMDAY **Последний день месяца**

Синтаксис:

E = eomday(<*год*>, <*месяц*>)

Описание:

Функция E = eomday(<*год*>, <*месяц*>) возвращает значения последнего дня месяца, причем аргументы <*год*> и <*месяц*> могут быть заданы в виде массивов.

Пример:

Определить все високосные годы в период с 1985-го по 2010 год.

```
y = 1985:2010;
E = eomday(y, 2*ones(length(y), 1)');
y(find(E==29))'
ans =
      1988
      1992
      1996
      2000
      2004
      2008
```

Сопутствующие функции: WEEKDAY, DATENUM, DATEVEC.

DATETICK **Выбор формата даты для разметки графических осей**

Синтаксис:

datetick(<*имя_оси*>, <*формат_даты*>)

Описание:

Команда datetick(<*имя_оси*>, <*формат_даты*>) размечает указанную ось в соответствии с выбранным форматом даты. Аргумент <*имя_оси*> может принимать одно из строковых значений: 'x', 'y' или 'z'; по умолчанию используется 'x'. Метки форматируются в соответствии со значением аргумента <*формат_даты*>, которое может быть задано либо числовым номером формата, либо строкой формата согласно приведенной ниже таблицы. Если этот аргумент отсутствует, то разметка реализуется в соответствии со значениями данных, размещаемых вдоль этой оси координат.

Для получения правильных результатов дата должна быть задана в числовом формате, соответствующем функции datenum.

Если включен режим сохранения текущего графика hold on, то будет выполнено преобразование меток, но они не будут выведены на график.

Функция datetick использует функцию datestr для преобразования данных из числового формата в строковый.

Формат даты		*Пример*
Номер	*Формат*	
0	'dd-mmm-yyyy HH:MM:SS'	01-Mar-1995 15:45:17
1	'dd-mmm-yyyy'	01-Mar-1995
2	'mm/dd/yy'	03/01/95
3	'mmm'	Mar
4	'm'	M
5	'mm'	3
6	'mm/dd'	03/01
7	'dd'	1
8	'ddd'	Wed
9	'd'	W
10	'yyyy'	1995
11	'yy'	95
12	'mmmyy'	Mar95
13	'HH:MM:SS'	15:45:17
14	'HH:MM:SS PM'	3:45:17 PM
15	'HH:MM'	15:45
16	'HH:MM PM'	3:45 PM
17	'QQ-YY'	Q1-96
18	'QQ'	Q1

Пример:

Данные основаны на переписи населения США в 1990 году.

```
t = (1900:10:1990)'; % Промежуток времени
p = [75.995 91.972 105.711 123.203 131.669 ...
    150.697 179.323 203.212 226.505 249.633]';  %Численность
plot(datenum(t, 1, 1), p) %Преобразовать годы в даты и построить график
datetick('x', 'yyyy') % Преобразовать разметку оси x  в формат 'yyyy'.
    grid
```

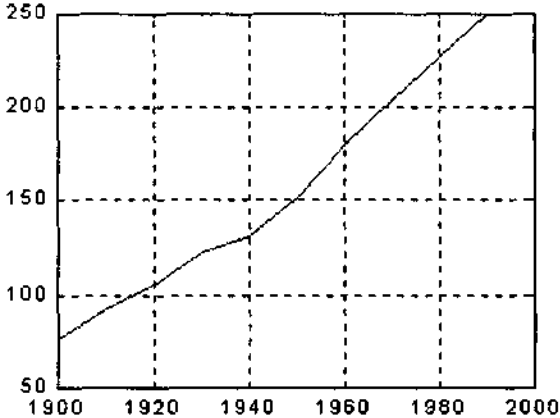


Рис. 4.1

Сопутствующие функции: DATESTR, DATENUM.

Интервалы времени

Для отсчета интервалов времени предназначены следующие функции:

Cputime	Время работы центрального процессора
Tic	Начало отсчета
Toc	Конец отсчета
Etime	Прошедшее время

CPUTIME **Подсчет процессорного времени**

Синтаксис:

cputime

Описание:

Функция cputime возвращает время работы процессора (в секундах), использованное системой MATLAB с момента ее запуска. Возвращаемая величина может превысить внутреннее представление. В этом случае отсчет времени начинается заново.

Пример:

Для учета процессорного времени, необходимого для решения пользовательской задачи, можно выполнить следующие операции:

```
t0 = cputime; surf(peaks(40)); e = cputime - t0
e =      11.54
```

Сопутствующие функции: ETIME, TIC, TOC, CLOCK.

TIC, TOC **Запуск и остановка таймера**

Синтаксис:

tic

последовательность операторов

toc

Описание:

Функции tic и toc используются в паре для запуска и остановки таймера с выводом его показания.

Пример:

```
tic,  surf(peaks(40)); toc
elapsed_time = 7.58
```

Сопутствующие функции: CLOCK, ETIME, CPUTIME.

ETIME **Определение временного интервала**

Синтаксис:

e = etime(t1, t0)

Описание:

Функция e = etime(t1, t0) возвращает длительность в секундах промежутка времени, задаваемого векторами t1 и t0. Каждый из этих векторов должен состоять из шести элементов в формате, соответствующем результатам выполнения функции clock:

T = [*год месяц день час минута секунда*].

Пример:

```
t0 = clock; surf(peaks(40)); etime(clock, t0)
ans = 8.68
```

Ограничения:

Приведенный фрагмент программы будет работать некорректно, если в текущий промежуток времени попадут границы месяца или года.

Сопутствующие функции: TIC, TOC, CLOCK, CPUTIME.

Функции истинности

ALL, ANY **Проверить элементы массива на логическое условие**

Синтаксис:

= all(x<*оператор_отношения*> *выражение*)

= all(X<*оператор_отношения*> *выражение*)

Y = all(X<*оператор_отношения*>*выражение*, <*размерность*>)

y = any(x<*оператор_отношения*> *выражение*)

y = any(X<*оператор_отношения*> *выражение*)

Y = any(X<*оператор_отношения*> *выражение*, <*размерность*>)

Описание:

Функция y = all(x<*оператор_отношения*> *выражение*), когда x - вектор, возвращает 1, если **все** элементы x удовлетворяют заданному условию, и 0, если **хотя бы один** элемент x не удовлетворяет этому условию.

Функция y = all(X), когда X - двумерный массив, возвращает вектор-строку из 0 и 1. Элемент вектора-строки равен 1, если **все** элементы столбца X(:, j) удовлетворяют заданному условию, и 0, если **хотя бы один** элемент столбца X(:, j) не удовлетворяет этому условию.

Функция Y = all(X, <*размерность*>), когда X - многомерный массив, обрабатывает двумерный массив, соответствующий указанной размерности. Если аргумент <*размерность*> не указан, по умолчанию обрабатывается первая размерность, не равная 1.

Функция y = any(x), когда x - вектор, возвращает 1, если **хотя бы один** элемент x удовлетворяет заданному условию, и 0, если **все** элементы x не удовлетворяют этому условию.

Функция y = any(X), когда X - двумерный массив, возвращает вектор-строку из 0 и 1. Элемент вектора-строки равен 1, если **хотя бы один** элемент X(:, j) удовлетворяет заданному условию, и 0, если **все** элементы X(:, j) не удовлетворяют этому условию.

Функция Y = any(X, <*размерность*>), когда X - многомерный массив, обрабатывает двумерный массив, соответствующий указанной размерности. Если аргумент <*размерность*> не указан, по умолчанию обрабатывается первая размерность, не равная 1.

Пример:

```
x = [0.53 0.67 0.01 0.38 0.07 0.42 0.69];
y = (x < 0.5)
y = 0   0   1   1   1   1   0
all(y)
 ans =  0
```

Рассмотрим трехмерный массив размера 3×3×2.

```
A(:,:,2) =
     1     1     1
     0     0     1
```

```
A(:,:,1) =
     1     1     1
     1     1     0
```

```
all(A, 3)
ans =
   1   1   1
   0   0   0
```

```
all(A, 2)
ans(:, :, 1) =
          1
                    0
ans(:, :, 2) =
          1
          0
all(A, 1)
ans(:, :, 1) =  1    1    0
ans(:, :, 2) =  0    0    1
```

Рассмотрим для того же многомерного массива использование оператора any:

```
any(A, 3)
ans =
   1    1    1
   1    1    1
any(A, 2)
ans(:, :, 1) =
          1
          1
ans(:, :, 2) =
          1
          1
any(A,1)
ans(:, :, 1) = 1    1    1
ans(:, :, 2) = 1    1    1
```

Сопутствующие операторы: &, |, ~.

EXIST **Проверка существования переменной, файла или функции**

Синтаксис:

e = exist('<*имя_элемента*>')

e = exist('<*имя_элемента*>', '<*тип_элемента*>')

Описание:

Функция e = exist('<*имя_элемента*>') возвращает значения согласно таблице:

e	*Элемент*
7	Каталог
6	P-файл
5	Встроенная функция MATLAB
4	MDL-файл
3	MEX-файл
2	M-файл или файл неизвестного типа
1	Переменная в рабочей области
0	Не является ни одним из перечисленных выше элементов

Если функции exist('<*имя_файла*>') или exist('<*имя_файла*>.***') возвращают значение 2, то это означает, что файл с таким именем принадлежит списку путей доступа системы MATLAB, но расширение этого файла не есть mdl, p или mex.

В качестве аргумента <*имя_элемента*> можно использовать частичный путь доступа.

Функция e = exist('<*имя_элемента*>', '<*тип_элемента*>') позволяет указать тип тестируемого элемента согласно следующей таблице:

Тип элемента	*Назначение*
var	Проверять только на принадлежность к переменным
builtin	Проверять только на принадлежность к встроенным функциям
file	Проверять только на принадлежность к файлам
dir'	Проверять только на принадлежность к каталогам

Пример:

Проверить, является ли функция eig встроенной или М-файлом.

```
e = exist('eig')
e =  5
```

Функция eig - встроенная функция.

Сопутствующие функции: WHICH, WHO, WHAT, DIR, HELP, LOOKFOR.

IS*

Проверка истинности

Синтаксис:

```
k = iscell(C)
k = iscellstr(S)
k = ischar(S)
k = isempty(A)
k = isequal(A, B, ...)
k = isfield(S, 'имя_поля')
TF = isfinite(A)
k = isglobal(<имя_переменной>)
TF = ishandle(H)
k = ishold
k = isieee
TF = isinf(A)
TF = isletter('строка')
k = islogical(A)
TF = isnan(A)
k = isnumeric(A)
k = isobject(A)
k = isppc
TF = isprime(A)
k = isreal(A)
TF = isspace('строка')
k = issparse(S)
k = isstruct(S)
k = isstudent
k = isunix
k = isvms
```

Описание:

Функция k = iscell(C) возвращает 1, если C - массив ячеек, и 0 - в остальных случаях.

Функция k = iscellstr(S) возвращает 1, если S - массив строковых ячеек, и 0 - в остальных случаях. Массив строковых ячеек - это такой массив ячеек, в котором каждая ячейка является массивом символов.

Функция k = ischar(S) возвращает 1, если S - массив символов, и 0 - в остальных случаях.

Функция k = isempty(A) возвращает 1, если A - пустой массив, и 0 - в остальных случаях. Пустым массивом называется массив, у которого размер хотя бы по одной из размерностей равен 0.

Функция k = isequal(A, B, ...) возвращает 1, если входные массивы одного типа, одинаковых размеров и содержат одинаковые данные, и 0 - в остальных случаях.

Функция k = isfield(S, '*имя_поля*') возвращает 1, если имя поля совпадает с именем одного из полей массива записей S, и 0 - в остальных случаях.

Функция TF = isfinite(A) возвращает массив того же размера, что и A, состоящий из 1 и 0. Элемент равен 1, когда соответствующий элемент массива имеет конечное значение, и 0 - когда встречается элемент типа Inf или NaN.

Функция k = isglobal(<*имя_переменной*>) возвращает 1, если переменная с таким именем объявлена глобальной, и 0 - если нет.

Функция TF = ishandle(H) возвращает массив того же размера, что и H, состоящий из 1 и 0. Элемент равен 1, когда соответствующий элемент массива является дескриптором, и 0 - если нет.

Функция k = ishold возвращает 1, если включена графическая поддержка, и 0 - если нет. Когда графическая поддержка включена, текущий график (объект figure) и свойства его осей (объект axes) сохраняются, так что последующие графические построения добавляются к этому графику. Это означает, что свойство NextPlot для обоих объектов имеет значение Add.

Функция k = isieee возвращает 1 для компьютеров с IEEE-арифметикой (IBM PC, Macintosh, рабочие станции под ОС UNIX), и 0 - для компьютеров без IEEE-арифметики (VAX, Cray).

Функция TF = isinf(A) возвращает массив того же размера, что и A, состоящий из 1 и 0. Элемент равен 1, когда соответствующий элемент массива равен +Inf или -Inf, и 0 - если нет.

Функция TF = isletter('*строка*') возвращает массив размера, совпадающего с длиной строки и состоящий из 1 и 0. Элемент равен 1, когда соответствующий элемент строки является буквой алфавита, и 0 - если нет.

Функция k = islogical(A) возвращает 1, если A - массив логических выражений, и 0 - если нет.

Функция TF = isnan(A) возвращает массив того же размера, что и A, состоящий из 1 и 0. Элемент равен 1, когда соответствующий элемент массива равен NaN, и 0 - если нет.

Функция k = isnumeric(A) возвращает 1, если A - числовой массив, и 0 - если нет. Массивы чисел удвоенной точности и разреженные матрицы являются числовыми массивами, а строки, массивы ячеек и массивы записей таковыми не являются.

Функция k = isobject(A) возвращает 1, если A - объект системы MATLAB, и 0 - если нет.

Функция k = isppc возвращает 1, если используемый компьютер Macintosh Power PC, и 0 - в остальных случаях.

Функция TF = isprime(A) возвращает массив того же размера, что и A, состоящий из 1 и 0. Элемент равен 1, когда соответствующий элемент массива простое число, и 0 - если нет.

Функция k = isreal(A) возвращает 1, если все элементы массива A - вещественные числа, и 0 - если A - нечисловой массив или если некоторые элементы имеют ненулевые мнимые части. Так как строки являются подклассом числового массива, функция isreal всегда возвращает 1, если входом является строка. Поскольку система MATLAB поддерживает комплексную арифметику, то в процессе вычислений могут появляться комплексные числа; в связи с этим следует быть внимательным при использовании функции isreal.

Функция TF = isspace('*строка*') возвращает массив, размер которого совпадает с длиной входной строки и состоящий из 1 и 0. Элемент равен 1, когда соответствующий элемент массива является незаполненным пространством в кодах ASCII, и 0 - если нет. Незаполненное пространство в кодах ASCII - это пробел, символы новой строки, возврата каретки, табуляции, прогона страницы.

Функция k = issparse(S) возвращает 1, если S - разреженный массив, и 0 - если нет.

Функция k = isstruct(S) возвращает 1, если S - массив записей, и 0 - если нет.

Функция k = isstudent возвращает 1, если используется студенческая версия системы MATLAB, и 0 - если коммерческая.

Функция k = isunix возвращает 1, если используется версия системы MATLAB для ОС UNIX, и 0 - если нет.

Функция k = isvms возвращает 1, если используется версия системы MATLAB для ОС VMS, и 0 - если нет.

Пример:

```
s = 'A1,B2,C3';
isletter(s)
ans = 1   0   0   1   0   0   1   0
B = rand(2, 2, 2);
B(:, :, :) = [ ];
isempty(B)
ans = 1

A = [ 1 0 1 0 1 0;
      0 1 0 1 0 0];
B = A; C = A;
isequal(A, B, C)
ans =  1
```

```
a = [–2 –1 0 1 2];
isfinite(1./a)
Warning: Divide by zero.
ans = 1   1   0   1   1

isinf(1./a)
Warning: Divide by zero.
ans = 0   0   1   0   0

isnan(1./a) =
 Warning: Divide by zero.
ans = 0   0   0   0   0

isfinite(0./a)
Warning: Divide by zero.
ans = 1   1   0   1   1

isinf(0./a)
Warning: Divide by zero.
ans = 0   0   0   0   0

isnan(0./a)
Warning: Divide by zero.
ans = 0   0   1   0   0
```

Сопутствующие функции: WHICH, WHO, WHAT, DIR, HELP, LOOKFOR.

Операторы побитовой обработки

BITGET **Определить значение бита**

Синтаксис:

c = bitget(a, <*бит*>)

Описание:

Функция c = bitget(a, <*бит*>) возвращает значение бита в позиции <*бит*> для двоичного представления неотрицательного числа a; значение <*бит*> для компьютеров с IEEE-арифметикой и чисел с плавающей точкой лежит в диапазоне от 1 до 52.

Пример:

```
num2str(bitget(14,1:52))
    ans = 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Сопутствующие функции: BITSET, BITAND, BITOR, BITXOR, BITCMP, BITSHIFT, BITMAX.

BITSET **Установить значение бита**

Синтаксис:

c = bitset(a, <*бит*>)

c = bitset(a, <*бит*>, <*значение_бита*>)

Описание:

Функция c = bitset(a, <*бит*>) устанавливает значение бита в позиции <*бит*> для двоичного представления неотрицательного числа a в 1; значение <*бит*> для компьютеров с IEEE-арифметикой и чисел с плавающей точкой лежит в диапазоне от 1 до 52.

Функция c = bitset(a, <*бит*>, <*значение_бита*>) устанавливает значение бита в позиции <*бит*> для двоичного представления неотрицательного числа a, равное 0 или 1 в зависимости от аргумента <*значение_бита*>.

Пример:

Преобразовать число 14 в число 30.

```
bitset(14, 5, 1)
ans =   30
num2str(bitget(c,1:52))
    ans = 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Сопутствующие функции: BITGET, BITAND, BITOR, BITXOR, BITCMP, BITSHIFT, BITMAX.

BITMAX **Максимальное целое число**

Синтаксис:

c = bitmax

Описание:

Функция c = bitmax вычисляет максимальное целое число для арифметики с плавающей точкой; двоичному представлению этого числа соответствуют все биты, равные 1. Для компьютеров с IEEE-арифметикой это значение равно 2^{53}-1.

Пример:

Преобразовать число 14 в число 30.

```
c = bitmax
num2str(bitget(c,1:52))
    c =  9.0072e+015
    ans = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Сопутствующие функции: BITAND, BITOR, BITXOR, BITCMP, BITSHIFT, BITSET, BITGET.

BITCMP **Побитовое n-разрядное дополнение**

Синтаксис:

c = bitcmp(a, n)

Описание:

Функция c = bitcmp(a, n) строит побитовое дополнение для двоичного представления неотрицательного числа a, используя n первых бит такого представления.

Пример:

Преобразовать число 30 в число 33.

```
num2str(bitget(30,1:52))
    ans = 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
c = bitcmp(30, 6)
c =   33
num2str(bitget(c,1:52))
    ans = 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Сопутствующие функции: BITAND, BITOR, BITXOR, BITSHIFT, BITSET, BITGET, BITMAX.

BITAND **Логическое умножение a ⊗ b**

Синтаксис:

c = bitand(a, b)

Описание:

Функция c = bitand(a, b) выполняет операцию логического умножения a AND b, или конъюнкции, над целыми числами a и b в диапазоне от 1 до bitmax.

Пример:

```
num2str(bitget(12, 1:4))
ans = 0 0 1 1
num2str(bitget(10, 1:4))
ans = 0 1 0 1
num2str(bitget(bitand(12, 10), 1:4))
ans = 0 0 0 1
```

Сопутствующие функции: BITOR, BITXOR, BITCMP, BITSHIFT, BITSET, BITGET, BITMAX.

BITOR **Логическое сложение a + b**

Синтаксис:

c = bitor(a, b)

Описание:

Функция c = bitor(a, b) выполняет операцию логического сложения a OR b, или слабой дизъюнкции, над целыми числами a и b в диапазоне от 1 до bitmax.

Пример:

```
num2str(bitget(12, 1:4))
ans = 0  0  1  1
num2str(bitget(10, 1:4))
ans = 0  1  0  1
num2str(bitget(bitor(12, 10), 1:4))
ans = 0  1  1  1
```

Сопутствующие функции: BITAND, BITXOR, BITSHIFT, BITCMP, BITSET, BITGET, BITMAX.

BITXOR **Сложение по модулю 2 a ⊕ b**

Синтаксис:

```
c = bitxor(a, b)
```

Описание:

Функция c = bitxor(a, b) выполняет операцию сложения по модулю 2 a XOR b, или строгой дизъюнкции, над целыми числами a и b в диапазоне от 1 до bitmax.

Пример:

```
num2str(bitget(12, 1:4))
ans = 0  0  1  1
num2str(bitget(10, 1:4))
ans = 0  1  0  1
num2str(bitget(bitxor(12, 10), 1:4))
ans = 0  1  1  0
```

Сопутствующие функции: BITAND, BITXOR, BITSHIFT, BITCMP, BITSET, BITGET, BITMAX.

BITSHIFT **Сложение по модулю 2 a ⊕ b**

Синтаксис:

```
c = bitshift(a, k, n)
c = bitshift(a, k)
```

Описание:

Функция c = bitshift(a, k, n) выполняет операцию сдвига на k бит. Это равносильно умножению на 2^k; если сдвиг приводит к переполнению n бит, то биты переполнения обнуляются.

Функция c = bitshift(a, k) выполняет операцию сдвига на k бит. Значение n по умолчанию равно 53.

Пример:

```
bitshift(3, 2, 4)
ans =    12
num2str(bitget(bitshift(3, 2, 4), 1:4))
```

```
ans = 0  0  1  1
bitshift(3, 2, 3)
ans =    4
num2str(bitget(bitshift(3, 2, 3), 1:4))
ans = 0  0  1  0
```

Сопутствующие функции: BITAND, BITOR, BITXOR, BITCMP, BITSET, BITGET, BITMAX.

Операторы обработки множеств

ISMEMBER **Выявление одинаковых элементов**

Синтаксис:

```
c = ismember(a, b)
c = ismember(A, B, 'rows')
```

Описание:

Функция c = ismember(a, b) формирует вектор C, элементами которого являются 0 и 1; при совпадении элементов множеств a и b присваивается 1, при несовпадении - 0. В качестве множеств могут быть векторы и массивы строковых ячеек.

Функция c = ismember(A, B, 'rows'), где A и B - массивы с одинаковым количеством столбцов, формирует вектор c, в котором содержится 1, если строки совпадают, и 0, если нет.

Пример:

```
set = [0 2 4 6 8 10 12 14 16 18 20];
a = reshape(1:5, [5 1])
a =
1
2
3
4
5
ismember(a, set)
ans =
0
1
0
1
0
```

Сопутствующие функции: UNIQUE, INTERSECT, SETDIFF, SETXOR, UNION.

Переопределяемые методы:

```
help cell/ismember.m
```

UNIQUE **Удаление из множества одинаковых элементов**

Синтаксис:

```
b = unique(a)
B = unique(A, 'rows')
[B, I, J] = unique(...)
```

Описание:

Функция b = unique(a) удаляет из множества a одинаковые элементы. В качестве множества могут быть вектор и массив строковых ячеек.

Функция B = unique(A, 'rows'), где A - массив, формирует массив B, из которого удалены одинаковые строки.

Функция [B, I, J] = unique(...) возвращает, кроме того, векторы индексов I и J, указывающие способ сортировки.

Пример:

```
a = [1 1 5 6 2 3 3 9 8 6 2 4]
a =  1 1 5 6 2 3 3 9 8 6 2 4
[b, i, j] = unique(a)
b = 1 2 3 4 5 6 8 9
i =  2 11 7 12 3 10 9 8
j =  1 1 5 6 2 3 3 8 7 6 2 4
a(i)
ans = 1 2 3 4 5 6 8 9
b(j)
ans = 1 1 5 6 2 3 3 9 8 6 2 4
```

Сопутствующие функции: UNION, INTERSECT, SETDIFF, SETXOR, ISMEMBER.

Переопределяемые методы:

help cell/unique.m

UNION **Объединение множеств**

Синтаксис:

```
c = union(a, b)
C = union(A, B, 'rows')
[C, IA, IB] = union(...)
```

Описание:

Функция c = union(a, b) формирует объединение множеств a и b в множество c без повторения одинаковых элементов. В качестве множеств могут быть векторы и массивы строковых ячеек.

Функция C = union(A, B, 'rows'), где A и B - массивы с одинаковым количеством столбцов, формирует массив C, в котором объединены строки без повторений, если они одинаковы.

Функция вида [C, IA, IB] = union(...) возвращает, кроме того, векторы индексов IA и IB, указывающие способ сортировки.

Пример:

```
a = [–1 0 2 4 6];
b = [–1 0 1 3];
[c, ia, ib] = union(a, b);
c = –1 0 1 2 3 4 6
ia =  3 4 5
ib =  1 2 3 4
```

Сопутствующие функции: UNIQUE, INTERSECT, SETDIFF, SETXOR, ISMEMBER.
Переопределяемые методы:

```
help cell/union.m
```

INTERSECT **Пересечение множеств**

Синтаксис:

```
c = intersect(a, b)
C = intersect(A, B, 'rows')
[C, IA, IB] = intersect(...)
```

Описание:

Функция c = intersect(a, b) формирует пересечение множеств a и b, отбирая только общие элементы. В качестве множеств могут быть векторы и массивы строковых ячеек.

Функция C = intersect(A, B, 'rows'), где A и B - массивы с одинаковым количеством столбцов, формирует массив C, в котором содержатся общие строки массивов A и B.

Функция вида [C, IA, IB] = intersect(...) возвращает, кроме того, векторы индексов IA и IB, указывающие способ сортировки.

Пример:

```
A = [1 2 3 6]; B = [1 2 3 4 6 10 20];
[c, ia ,ib] = intersect(A, B);
disp([c; ia; ib])
1 2 3 6
1 2 3 4
1 2 3 5
```

Сопутствующие функции: UNIQUE, UNION, SETDIFF, SETXOR, ISMEMBER.
Переопределяемые методы:

```
help cell/intersect.m
```

SETDIFF **Разность множеств**

Синтаксис:

```
c = setdiff(a, b)
C = setdiff(A, B, 'rows')
[C, IA, IB] = setdiff(...)
```

Описание:

Функция c = setdiff(a, b) формирует множество C, элементами которого являются несовпадающие элементы множеств a и b. В качестве множеств могут быть векторы и массивы строковых ячеек.

Функция C = setdiff(A, B, 'rows'), где A и B - массивы с одинаковым количеством столбцов, формирует массив C, в котором содержатся несовпадающие строки массивов A и B.

Функция вида [C, IA, IB] = setdiff(...) возвращает, кроме того, векторы индексов IA и IB, указывающие способ сортировки.

Пример:

```
A = magic(5);
B = magic(4);
[c, i] = setdiff(A, B);
c' = 17 18 19 20 21 22 23 24 25
i' =  1 10 14 18 19 23 2 6 15
```

Сопутствующие функции: UNIQUE, UNION, INTERSECT, SETXOR, ISMEMBER.
Переопределяемые методы:

```
help cell/setdiff.m
```

SETXOR **Разность множеств**

Синтаксис:

```
c = setxor(a, b)
C = setxor(A, B, 'rows')
[C, IA, IB] = setxor(...)
```

Описание:

Функция c = setxor(a, b) формирует множество C, элементами которого являются те элементы, которые не принадлежат пересечению множеств a и b. В качестве множеств могут быть векторы и массивы строковых ячеек.

Функция C = setxor(A, B, 'rows'), где A и B - массивы с одинаковым количеством столбцов, формирует массив C, в котором содержатся строки, не принадлежащие пересечению массивов A и B.

Функция вида [C, IA, IB] = setxor(...) возвращает, кроме того, векторы индексов IA и IB, указывающие способ сортировки.

Пример:

```
a = [-1 0 1 Inf -Inf NaN];
b = [-2 pi 0 Inf];
c = setxor(a, b)
c = -Inf -2.0000 -1.0000 1.0000 3.1416 NaN
```

Сопутствующие функции: UNIQUE, UNION, INTERSECT, SETDIFF, ISMEMBER.
Переопределяемые методы:

```
help cell/setxor.m
```

5. ПРОГРАММИРОВАНИЕ И ОТЛАДКА

Система MATLAB обычно работает в режиме интерпретации команд и операторов: они вводятся в ходе сеанса в командной строке, а MATLAB выполняет их немедленную обработку и выдает вычисленный результат. Однако в MATLAB есть возможность обработки заранее подготовленной последовательности команд и операторов, записанной в виде файла.

Сценарии, функции и переменные

Файлы, содержащие команды и операторы MATLAB, называются *М-файлами*. Существует два типа М-файлов: *М-сценарии* и *М-функции* - со следующими характеристиками:

М-сценарий	*М-функция*
Не допускает входных и выходных аргументов	Допускает входные и выходные аргументы
Оперирует с данными из рабочей области	По умолчанию внутренние переменные являются локальными по отношению к функции
Предназначен для автоматизации последовательности шагов, которые нужно выполнять много раз	Предназначена для расширения возможностей языка MATLAB (библиотеки функций, пакеты прикладных программ)

SCRIPT **Заголовок сценария**

Синтаксис:

script

Описание:

Команда script задает заголовок сценария и записывается в первой строке. Сценарий (Script-файл) - это внешний файл, содержащий последовательность команд, операторов и функций системы MATLAB. После ввода имени такого файла начинается последовательная интерпретация его операторов. Посредством изменения заголовка сценарий может быть преобразован в процедуру-функцию.

Сценарии являются самым простым типом М-файла - у них нет входных и выходных аргументов. Они позволяют автоматизировать выполнение последовательностей команд, которые в ином случае должны были бы многократно вводиться из командной строки. Сценарии оперируют данными из рабочей области и могут создавать новые данные для последующей обработки в этом файле. Хотя М-сценарии не возвращают выходных аргументов,

они сохраняют все переменные, которые используются в сценарии, в рабочей области; после выполнения сценария они могут быть использованы для дальнейших вычислений.

Любой текст, которому предшествует символ %, является комментарием, и он может либо присутствовать в сценарии в виде отдельной строки, либо следовать после любой из команд.

Имена М-сценариев не разрешается использовать в качестве операндов в выражениях или в качестве аргументов функций.

Пример:

Следующие операторы вычисляют радиус-вектор rho для различных тригонометрических функций от угла theta и строят последовательность графиков в полярных координатах.

Строка комментария

Вычисления

Команды графического вывода

```
% M-file peetals - сценарий построения лепесткового графика
theta = -pi:0.01:pi;
rho(1, :) = 2*sin(5*theta).^2;
rho(2, :) = cos(10*theta).^3;
rho(3, :) = sin(theta).^2;
rho(4, :) = 5*cos(3.5*theta).^3;
for i = 1:4
      polar (theta, rho(i, :))
      pause
end
```

Создайте М-файл petals.m, вводя указанные выше операторы. Этот файл является М-сценарием. Ввод команды petals в командной строке системы MATLAB вызывает выполнение этого сценария.

После того как сценарий отобразит первый график, нажмите клавишу Return, чтобы перейти к следующему графику. В сценарии отсутствуют входные и выходные аргументы; программа petals сама создает переменные, которые сохраняются в рабочей области системы MATLAB. Когда выполнение завершено, переменные (i, theta и rho) остаются в рабочей области. Для того чтобы увидеть их перечень, введите команду whos.

Сопутствующие команды и функции: ECHO, FUNCTION, TYPE.

FUNCTION **Заголовок процедуры-функции**

Синтаксис:

function[<*список выходных переменных*>] = <*имя функции*>(<*список входных переменных*>)

Описание:

Если первая строка М-файла начинается с определения

function[<*список выходных переменных*>] = <*имя функции*>(<*список входных переменных*>)

то этот файл представляет собой М-функцию. Эти файлы расширяют возможности системы благодаря добавлению новых функций, самостоятельно написанных пользователем на языке MATLAB.

М-функция включает следующие компоненты:

Строка определения функции
Первая строка комментария
Комментарий
Тело функции

```
function f = fact (n)
% FACT Вычисление факториала.
% fact(n) возвращает n! - факториал
% числа n.
% Вычислить fact (n) = prod(1:n).
f = prod(1:n);
```

- **строка определения функции** задает имя, количество и порядок следования входных и выходных аргументов;
- **первая строка комментария** определяет назначение функции. Она выводится на экран с помощью команд lookfor или help<*имя каталога*>;
- **комментарий** выводится на экран вместе с первой строкой при использовании команды help<*имя функции*>;
- **тело функции** - это программный код, который реализует вычисления и присваивает значения выходным аргументам.

М-функции используются так же, как обычные встроенные функции системы MATLAB. Имя М-функции можно применять при записи арифметических выражений.

Возможность добавления к системе MATLAB новых функций обеспечивает ей важное свойство расширяемости. Новые функции могут использовать как встроенные функции системы, так и функции, ранее написанные пользователем. Новые функции добавляются к словарю системы MATLAB и становятся доступными наряду со встроенными функциями. Эти функции оформляются в виде текстовых файлов с расширением .m.

Функция в системе MATLAB не имеет никаких отличительных признаков, и при обращении к ней MATLAB просто ищет файл с соответствующим именем, просматривая все пути доступа, определенные переменной MATLABPATH. Когда файл найден, он проходит процедуру компиляции, размещается в оперативной памяти и только после этого начинает выполняться. В этом также отличие функции от Script-файла, который выполняется только в режиме интерпретации. Для удаления откомпилированной функции из памяти используется команда clear <*имя функции*>.

Если при выполнении функции включен режим echo, то функция тоже будет выполняться в режиме интерпретации и на терминал будет выводиться результат выполнения каждого оператора.

М-функции являются файлами, которые допускают наличие входных и выходных аргументов. Они работают с переменными в пределах собственной рабочей области, отличной от рабочей области системы MATLAB.

Начиная с версии MATLAB 5 М-файлы могут содержать коды для более чем одной функции. Первая функция в файле - это *основная функция*, вызываемая по имени М-файла. Другие функции внутри файла - это *подфункции*, которые являются видимыми только для *основной функции* и других *подфункций* этого же файла.

Каждая подфункция имеет свой собственный заголовок. Подфункции следуют друг за другом непрерывно. Подфункции могут вызываться в любом порядке, в то время как основная функция выполняется первой.

Основная функция

```
function [avg, med] = newstats (u)
% NEWSTATS Находит среднее значение и медиану для
элементов вектора u, используя встроенные функции.
n = length(u);
avg = mean(u,n);
med = median(u,n);
```

Подфункция 1

```
function m = mean(v,n)
% Вычислить среднее.
a = sum(v)/n;
```

Подфункция 2

```
function m = median(v,n)
% Вычислить медиану.
w = sort(v);
if rem(n,2) == 1
   m = w ((n + 1) /2);
else
   m = (w (n/2) + w (n/2 + 1)) /2;
end
```

Подфункции mean и median вычисляют среднее и медиану входного списка. Основная функция newstats определяет длину списка и вызывает подфункции, передавая им длину списка n. Функции внутри одного и того же М-файла не могут обращаться к одним и тем же переменным, если они не объявлены глобальными переменными внутри соответствующих функций или не переданы им в качестве параметров. Следует иметь в виду, что справка help может видеть только основную функцию и не видит подфункций.

Когда приходит вызов функции из М-файла, то MATLAB в первую очередь проверяет, не является ли эта функция подфункцией. Поскольку первой проверяется наличие подфункций, то можно в качестве имени подфункции использовать имена функций системы MATLAB.

Пример:

Функция average - это достаточно простой М-файл, который вычисляет среднее значение элементов вектора:

```
function y = average (x)
% AVERAGE Среднее значение элементов вектора.
% AVERAGE(X), где X - вектор. Вычисляет среднее значение
элементов вектора.
%Если входной аргумент не является вектором, генерируется
ошибка.

[m,n] = размер(x);
if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))
      error('Input must be a vector')
end

y =sum(x)/length(x);          %Собственно вычисление
```

Введите эти команды в М-файл с именем average. Функция average допускает единственный входной и единственный выходной аргументы. Для того чтобы вызвать функцию average, введите следующие операторы:

```
z = 1:99;
average(z)
ans =    50
```

Сопутствующие функции: NARGIN, NARGOUT, PCODE, VARARGIN, VARARGOUT, WHAT.

GLOBAL **Определение глобальных переменных**

Синтаксис:

global *имя_1 имя_2 ...*

Описание:

Команда global *имя_1 имя_2 имя_3* определяет переменные с именами имя_1, имя_2, имя_3 как *глобальные.*

Как правило, каждая М-функция использует локальные переменные, которые изолированы от переменных других функций и рабочей области. Однако если несколько функций и сценарий объявляют некоторую переменную глобальной, то в этом случае все модули используют одну и ту же копию этой переменной. Присваивание значения такой переменной возможно в любой из функций, где эта переменная объявлена глобальной. Если глобальная переменная в момент объявления не существует, то ей присваивается значение пустого массива. Обычно для имен глобальных переменных используют длинные названия и заглавные буквы, но это необязательно.

Ошибка при объявлении переменной глобальной возникает в следующих случаях:

если существует переменная с тем же именем в рабочей области;

если переменная с этим именем была ранее вызвана в некотором М-файле.

Замечание:

Рекомендуется использовать команду clear global variable для удаления глобальных переменных из специальной рабочей области, предназначенной для размещения глобальных переменных.

Рекомендуется использовать команду clear variable, чтобы удалить глобальную переменную из рабочей области, не изменяя значения самой глобальной переменной.

Чтобы использовать глобальную переменную при повторном вызове, объявите ее, используйте, а затем удалите из рабочей области. Это позволяет избежать необходимости объявлять глобальную переменную после того, как на нее была сделана ссылка. Этот прием иллюстрируется следующим фрагментом:

```
uicontrol('style', 'pushbutton', 'CallBack', ...
'global MY_GLOBAL, disp(MY_GLOBAL), MY_GLOBAL = MY_GLOBAL+1,
clear MY_GLOBAL',...
'string', 'count')
```

Пример:

Рассмотрим программы для функций tic и toc (некоторые комментарии сокращены), которые манипулируют таймером. Глобальная переменная TICTOC используется обеими функциями, но это невидимо в основной рабочей области или в рабочих областях других функций, в которых не объявлена эта глобальная переменная.

```
function tic
% TIC Стартовать таймер.
% TIC; <операторы>; TOC
% выводит на терминал затраченное время.
% Сопутствующие функции: TOC, CLOCK.

global TICTOC
TICTOC = clock;

function t = toc
% TOC Зафиксировать истраченное время.
% TOC выводит на терминал затраченное время с момента вызова TIC.
% t = TOC; выводит затраченное время в виде переменной t.
% Сопутствующие функции: TIC, ETIME.

global TICTOC
if nargout < 1
elapsed_time = etime(clock, TICTOC)
else
t = etime(clock, TICTOC);
end
```

Сопутствующие функции: CLEAR, ISGLOBAL, WHO.

PERSISTENT **Определение сохраняемых переменных**

Синтаксис:

persistent *имя_1 имя_2 ...*

Описание:

Команда persistent *имя_1 имя_2 имя_3* определяет переменные с именами имя_1, имя_2, имя_3 как *сохраняемые*. Функция может быть использована только внутри некоторой М-функции для того, чтобы сохранить значения для одной или нескольких переменных между повторными вызовами этой функции.

Сохраняемые переменные удаляются, когда М-функция удаляется из памяти или когда в нее вносятся изменения.

Чтобы сохранить М-функцию в памяти до окончания сеанса работы, следует применить функцию mlock.

Если сохраняемая переменная в момент объявления не существует, то ей присваивается значение пустого массива.

Ошибка при объявлении переменной сохраняемой возникает в том случае, когда существует переменная с тем же именем в рабочей области.

Обычно для имен сохраняемых переменных используют длинные названия и заглавные буквы, но это необязательно.

Сопутствующие команды: CLEAR, GLOBAL, MISLOCKED, MLOCK, MUNLOCK.

MFILENAME **Имя последнего вызванного М-файла**

Синтаксис:

mfilename

Описание:

Команда mfilename возвращает строку с именем последнего вызванного М-файла. Когда вызов реализован внутри М-файла, команда возвращает имя этого М-файла, что позволяет определить его имя, даже если оно было изменено.

При использовании из командной строки команда mfilename возвращает пустой массив.

Для получения имен функций, вызванных М-файлом, следует применять команду dbstack с выходным аргументом.

Сопутствующие команды: DBSTACK, FUNCTION, NARGIN, NARGOUT, INPUTNAME.

LISTS **Определение списков значений**

Синтаксис:

help lists

Описание:

Команда help lists выводит на экран комментарий к определению и использованию списков значений при работе с массивами записей и массивами ячеек.

Извлечение множественных данных из массивов записей и массивов ячеек осуществляется с помощью *списков значений.*

Список значений для *массива записей* - это объединение одноименных полей S.name = [S(1).name S(2).name ... S(end).name].

Список значений для *массива ячеек* - это объединение ячеек C{:} = {C{1} C{2} ... C{end}}.

Конструкции вида S(m:n).name, C{m:n} также представляют собой списки значений.

Списки значений используются в следующих случаях:

- в командной строке для вывода значений на экран - S.name, C{:};
- при вызове М-функций - myfun(x, y, S.name), myfun(x, y, C{:});
- в операциях конкатенации - [S.name], [C{:}];
- в списках выходных аргументов функции - [S.name] = myfun, [C{:}] = myfun;
- как составляющие массива ячеек - {S.name}, {C{:}}.

Пример:

Рассмотрим некоторые примеры использования списков значений:

формирование массива ячеек:

```
C = {1 2 3 4}
C =      [1]     [2]     [3]     [4]
```

преобразование массива ячеек в числовой массив:

```
A = [C{:}]
A =      1       2       3       4
```

преобразование массива ячеек в трехмерный массив:

```
B = cat(3, C{:})
B(:, :, 1) =      1
B(:, :, 2) =      2
B(:, :, 3) =      3
B(:, :, 4) =      4
```

присвоение значений одному из полей массива записей:

```
[S(1:3).FIELD] = deal(5)
S =
3x1 struct array with fields:
    FIELD
S(:).FIELD
ans =      5
ans =      5
ans =      5
```

В результате выполненных операций были сформированы следующие массивы:

```
whos
  Name       Size          Bytes  Class
  A          1x4              32  double array
  B          1x1x4            32  double array
  C          1x4             400  cell array
  S          3x1             332  struct array
Grand total is 22 elements using 796 bytes
Общее количество элементов - 22; используют 796 байт
```

Сопутствующие функции: CAT, CELL2STRUCT, DEAL, NUM2CELL, STRUCT2CELL, VARARGIN, VARARGOUT.

MLOCK **Запретить удаление файла**

Синтаксис:

mlock

mlock(<*имя_М-файла*>)

Описание:

Команда mlock запрещает удаление текущего исполняемого М-файла.

Команда mlock(<*имя_М-файла*>) запрещает удаление М-файла с указанным именем с помощью команды clear до окончания сеанса работы.

Для возвращения М-файла в нормальное состояние, которое допускает его удаление, следует воспользоваться командами munlock или munlock(<*имя_М-файла*>).

Сопутствующие команды: MUNLOCK, MISLOCKED.

MUNLOCK **Снятие запрета на удаление файла**

Синтаксис:

munlock

munlock(<*имя_М-файла*>)

Описание:

Команда munlock снимает запрет на удаление текущего исполняемого М-файла.

Команда munlock(<*имя_М-файла*>) снимает запрет на удаление М-файла с указанным именем.

Обращение к командам группы munlock может потребоваться только в том случае, если использовалась команды группы mlock.

Сопутствующие команды: MLOCK, MISLOCKED.

MISLOCKED **Проверка запрета на удаление файла**

Синтаксис:

mislocked
mislocked(<*имя_М-файла*>)

Описание:

Команда mislocked возвращает 1, если существует запрет на удаление текущего исполняемого М-файла, и 0 - в ином случае.

Команда mislocked (<*имя_М-файла*>) возвращает 1, если существует запрет на удаление М-файла с указанным именем, и 0 - в ином случае.

Сопутствующие команды: MLOCK, MUNLOCK.

Вычисление и выполнение

Возможность интерпретации и исполнения символьных последовательностей, записанных в виде строковых выражений, придает языку MATLAB дополнительную мощь и гибкость. Это позволяет конструировать символьные последовательности в зависимости от хода решения задачи и таким образом придать сценарию решения задачи элементы искусственного интеллекта. Кроме того, открывается возможность обращаться по имени к ранее написанным функциям и вызывать их в зависимости от ситуации.

EVAL **Вычисление строковых выражений**

Синтаксис:

x = eval('<*выражение*>')
[x1, x2, x3, ...] = eval('<*выражение*>')
eval(string, catchstring)

Описание:

Функция x = eval('<*выражение*>') возвращает результат выполнения выражения, записанного в виде строки символов. Строка формируется путем объединения подстрок и переменных внутри квадратных скобок.

Функция [x1, x2, x3, ...] = eval('<*выражение*>') возвращает результат в виде отдельных переменных. Эта конструкция предпочтительнее eval('[x1, x2, x3, ...] = <*выражение*>'), поскольку в последней скрыта часть информации от синтаксического анализатора системы MATLAB и это может привести к непредсказуемому результату.

Конструкция eval('<*выражение*>', catchstring) позволяет выявлять ошибки при вычислении строкового выражения. Если возникла ошибка, то перед завершением команды формируется переменная catchstring. После этого можно с помощью функции lasterr получить сообщение об ошибке.

Пример:

Следующий фрагмент программы показывает, каким образом с помощью команды load можно загрузить 10 последовательно пронумерованных файлов с именами mydata.i:

```
fname = 'mydata';
for I=1:10
   eval(['load ', fname.int2str(i)])
end
```

Здесь int2str - функция преобразования числового значения в строковое.

Следующий программный код позволяет сформировать матрицу Гильберта порядка n:

```
t = '1/(i + j-1)';
n = 4;
for i = 1:n
     for j = 1:n
           G(i, j) = eval(t);
     end
end
format rational
G
G =
      1          1/2          1/3          1/4
     1/2         1/3          1/4          1/5
     1/3         1/4          1/5          1/6
     1/4         1/5          1/6          1/7
```

В следующем примере осуществляется выбор файла из списка и его выполнение. Обратите внимание, что все строки списка должны иметь одинаковую длину:

```
D = ['odedemo '
     'quaddemo'
     'zerodemo'
     'fitdemo '];
n = input('Укажите номер файла в списке:');
eval(D(n,:))
```

Сопутствующие функции: FEVAL, LASTERR.

FEVAL **Вычисление функции по заданному имени**

Синтаксис:

feval('<*имя_функции*>', x1, ..., xn)

[y1, ..., yk] = feval('<*имя_функции*>', x1, ..., xn)

Описание:

Функция feval('<*имя_функции*>', x1, ..., xn) передает аргументы (x1, ..., xn) вызываемой функции.

Функция [y1, ..., yk] = feval('<*имя_функции*>', x1, ..., xn) возвращает множество выходных переменных (y1, ..., yk).

Внутри методов, которые переопределяют встроенные функции, следует использовать конструкцию вида builtin(<*имя_функции*>, ...), чтобы выполнить исходную функцию.

Обычно функция feval применяется для вычисления функций, которые являются внешними, а их значение требуется перевычислять на каждом шаге в теле другой процедуры, например при интегрировании систем ОДУ или минимизации функции.

Пример:

Интегрирование дифференциального уравнения Ван дер Поля.

Функция вычисления правой части уравнения Ван дер Поля оформляется в виде внешней функции вида

```
function xdot = vdpol(t, x)
xdot = [x(1).*(1 - x(2).^2) - x(2); x(1)]
```

Для интегрирования системы ОДУ методом Рунге - Кутты предназначена функция ode23, в теле которой присутствуют следующие операторы, которые используют функцию vdpol, передаваемую в качестве входного аргумента tpfun процедуры ode23:

```
s1 = feval(ypfun, t, y); s1 = s1(:);
s2 = feval(ypfun, t + h, y + h * s1); s2 = s2(:);
s3 = feval(ypfun, t + h/2, y + h * (s1+s2)/4); s3 = s3(:);
```

Тогда для интегрирования уравнения Ван дер Поля требуется следующий вызов функции ode23:

```
t0 = 0; tf = 20; % Задание интервала времени интегрирования
x0 = [0 0.25]'; % Задание начальных условий
[t, x] = ode23('vdpol', t0, tf, x0);
```

Сопутствующие функции: ASSIGNIN, BUILTIN, EVAL, EVALIN.
Переопределяемые методы:

help inline/feval.m

BUILTIN **Выполнить исходный код функции внутри метода**

Синтаксис:

builtin('<*имя_функции*>', x1, ..., xn)

[y1, ..., yk] = builtin('<*имя_функции*>', x1, ..., xn)

Описание:

Функции builtin('<*имя_функции*>', x1, ..., xn) и [y1, ..., yk] = builtin('<*имя_функции*>', x1, ..., xn) используются внутри методов, чтобы выполнить исходный код некоторой функции, которая была переопределена в этом методе.

Функция [y1, ..., yk] = builtin('*<имя_функции>*', x1, ..., xn) возвращает множество выходных переменных (y1, ..., yk).

Сопутствующие функции: FEVAL.

ASSIGNIN **Присвоить значение переменной из указанной рабочей области**

Синтаксис:

assignin(ws, '*<имя_переменной>*', '*<значение>*')

Описание:

Функция assignin(ws, '*<имя_переменной>*', '*<значение>*') присваивает переменной с указанным именем из рабочей области ws заданное значение. Если переменной с таким именем не существует, она создается. Переменная ws может принимать два значения 'base' (основная рабочая область) и 'caller' (рабочая область вызванной функции).

Пример:

М-функция sqpi присваивает переменной var имя, задаваемое пользователем, в рабочей области М-функции и этой же переменной var присваивает значение sqrt(pi) в основной рабочей области.

```
function sqpi
var = inputdlg('Введите имя переменной', 'Assignin', 1, {'A'})
assignin('base', 'var', sqrt(pi))
```

```
sqpi
```

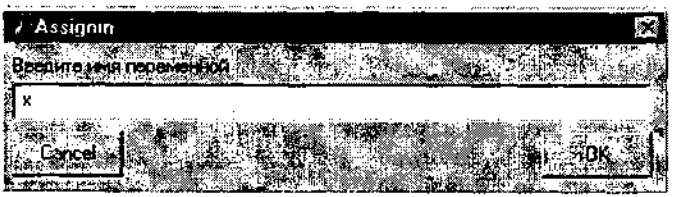


Рис. 5.1

```
var = 'x'
```

В основной рабочей области:

```
var
var = 1.7725
```

Сопутствующие функции: EVALIN.

EVALIN **Выполнить вычисления в указанной рабочей области**

Синтаксис:

evalin(ws, '*<выражение>*')

[x1, x2, x3, ...] = evalin(ws, '*<выражение>*')

evalin(ws, 'try', 'catch')

Описание:

Функция evalin(ws, '<*выражение*>') вычисляет строковое выражение с использованием переменных рабочей области, которая определяется переменной ws; эта переменная может принимать два значения 'base' (основная рабочая область) и 'caller' (рабочая область вызванной функции).

Функция [x1, x2, x3, ...] = evalin(ws, '<*выражение*>') возвращает результат в виде отдельных переменных.

Функция evalin(ws, 'try', 'catch') позволяет проверить правильность выражения 'try' и, если оно ошибочно в контексте рабочей области ws, сформировать сообщение 'catch'.

Функция evalin использует значения переменных из другой рабочей области, в то время как функция assignin позволяет изменять значения переменных в другой рабочей области.

Функцию evalin нельзя использовать рекурсивно.

Сопутствующие функции: ASSIGNIN, EVAL.

RUN **Выполнить М-сценарий**

Синтаксис:

run <*имя_М-сценария*>

Описание

Команда run <*имя_М-сценария*> позволяет запускать указанный М-сценарий из любого каталога, если в качестве имени указан полный путь доступа. Исполнение сценария будет реализовано в контексте текущей рабочей области. Это избавляет от необходимости использовать команды cd и addpath, которые приходилось применять в предшествующих версиях системы MATLAB, чтобы настроить требуемые пути доступа.

Сопутствующие функции: CD, ADDPATH.

Управление выполнением программ

Существует 4 основных оператора управления последовательностью исполнения инструкций:

- оператор цикла for выполняет группу инструкций фиксированное число раз;
- оператор условия while выполняет группу инструкций неопределенное число раз в соответствии с некоторым логическим условием завершения;
- оператор условия if в сочетании с операторами else и elseif выполняет группу инструкций в соответствии с некоторыми логическими условиями;
- оператор переключения switch в сочетании с операторами case и otherwise выполняет различные группы инструкций в зависимости от значения некоторого логического условия.

Все операторы управления включают оператор end, чтобы указать конец блока, в котором действует этот оператор управления.

Операторы организации циклов

FOR...END **Оператор цикла с определенным числом операций**

Синтаксис:

for <*переменная цикла*> = <*начальное значение*>:<*приращение*>:<*конечное значение*>
инструкции
end
for <*переменная цикла*> = A
инструкции
end

Описание:

Оператор цикла
for <*переменная цикла*> = <*начальное значение*>:<*приращение*>:<*конечное значение*>
инструкции
end
выполняет инструкцию или группу инструкций предопределенное число раз. По умолчанию приращение равно 1. Можно задавать любое приращение, в том числе отрицательное. Для положительных индексов выполнение завершается, когда значение индекса превышает <*конечное значение*>; для отрицательных приращений выполнение завершается, когда значение индекса становится меньше чем <*конечное значение*>.

Оператор цикла
for i = A
инструкции
end
определяет переменную цикла i как вектор A(:, k). Для первого шага цикла k равно 1; для второго - k равно 2 и т. д., пока k не достигнет значения n. То есть цикл выполняется столько раз, сколько столбцов в матрице A. Для каждого шага i - это вектор, содержащий один из столбцов массива A.

Пример:

Этот цикл выполняется 5 раз:
```
for i = 2:6
   x(i) = 2*x(i-1);
end
```
Допустимы вложенные циклы типа
```
for i = 1:m
   for j = 1:n
      A(i,j) = 1/(i + j - 1);
   end
end
```

Допустимы циклы с векторной переменной; цикл

```
x = [ ];
for v = [0 2 3 1]
  x = [ x  2^v]
end
```

формирует вектор, элементы которого являются степенями $2^{v(i)}$.

Сопутствующие операторы и функции: BREAK, END, IF, RETURN, SWITCH, WHILE.

WHILE ... END **Оператор цикла с неопределенным числом операций**

Синтаксис:

while *<логическое выражение>*
    <операторы>
end,

Описание:

Оператор цикла с неопределенным числом операций while ... end многократно выполняет инструкцию или группу инструкций, пока логическое выражение *истинно*.

Если выражение использует массив, то все его элементы должны быть истинны, чтобы выполнение продолжалось. Чтобы привести матрицу к скалярному значению, следует использовать функции any и all.

Логическое выражение имеет форму:

выражение <оператор отношения> выражение

оператор отношения: ==, <=, >=, <, >, ~

Пример:

Найти наименьшее число n, значение факториала которого записывается числом, содержащим 100 знаков.

```
n = 1;
while prod(1:n) < 1.e100, n = n+1; end
```

Определить машинную точность macheps, то есть такое наименьшее число, для которого еще выполняется условие 1+ macheps > 1:

```
eps = 1;
while (1 + eps) > 1
   eps = eps/2;
end
macheps = eps*2
```

Сопутствующие операторы и функции: ALL, ANY, BREAK, END, FOR, IF, RETURN, SWITCH.

Условные выражения

Структура условных выражений в языке MATLAB близка к обычно применяемой в языках программирования, но в них не используется оператор THEN.

Для организации условных выражений используются 4 оператора: if, else, elseif, end. Как и в операторах цикла, каждый условный оператор if должен заканчиваться оператором end.

IF...ELSE...ELSEIF...END **Оператор условия**

Синтаксис:

```
if логическое_выражение
   инструкции
end
```

```
if логическое_выражение
   инструкции
else
   инструкции
end
```

```
if логическое_выражение
   инструкции
elseif
логическое_выражение
   инструкции
else
   инструкции
end
```

Описание:

Оператор условия if ... end вычисляет некоторое логическое выражение и выполняет соответствующую группу инструкций в зависимости от значения этого выражения. Если логическое выражение *истинно*, то MATLAB выполнит все инструкции между if и end, а затем продолжит выполнение программы в строке после end. Если условие *ложно*, то MATLAB пропускает все утверждения между if и end и продолжает выполнение в строке после end.

Операторы if ... else ... end и if ... elseif ... end создают дополнительные ветвления внутри тела оператора if:

- оператор else не содержит логического условия. Инструкции, связанные с ним, выполняются, если предшествующий оператор if (и возможно, elseif) *ложны*;
- оператор elseif содержит логическое условие, которое вычисляется, если предшествующий оператор if (и возможно, elseif) *ложны*. Инструкции, связанные с оператором elseif, выполняются, если соответствующее логическое условие *истинно*. Оператор elseif может многократно использоваться внутри оператора условия if.

Пример:

```
if rem(a, 2) == 0
    disp('a четно')
    b = a/2;
end
```

Если логическое условие включает переменную, не являющуюся скаляром, то утверждение будет *истинным*, если *все* элементы *отличны от нуля*. Пусть задана матрица X; запишем следующий оператор условия:

```
if X
 инструкции
end
```

Этот оператор равносилен следующему:

```
if all(X(:))
 инструкции
end
```

Если в операторе if условное выражение является пустым массивом, то такое условие *ложно*. То есть оператор условия вида

```
if A
  S1
else
  S0
end
```

выполнит инструкции S0 только тогда, когда A - *пустой массив*.

Формирование трехдиагональной матрицы:

```
n = 4;
for i = 1:n
  for j = 1:n
    if i == j
      A(i, j) = 2;
    elseif abs(i-j) == 1
      A(i, j) = 1;
    else
      A(i, j) = 0;
    end
  end
end
A
A =
   2   1   0   0
   1   2   1   0
   0   1   2   1
   0   0   1   2
```

Сопутствующие функции и операторы: BREAK, ELSE, END, FOR, RETURN, SWITCH, WHILE.

BREAK **Прерывание выполнения цикла**

Синтаксис:

```
break
```

Описание:

Функция break прерывает выполнение циклов for и while. В случае вложенных циклов прерывание возможно только из самого внутреннего цикла.

Пример:

Цикл while ... end будет выполняться до тех пор, пока не будет введено нулевое или отрицательное значение переменной n:

```
while 1
   n = input('Введите n. Прекращение ввода n <= 0. n = ')
   if n <= 0, break, end
   r = rank(magic(n))
end
```

Сопутствующие операторы: END, ERROR, FOR, IF, RETURN, SWITCH, WHILE.

SWITCH...CASE...OTHERWISE...END **Оператор переключения**

Синтаксис:

```
switch <выражение>
        % выражение - это обязательно скаляр или строка
case <значение_1>
        инструкции
       % выполняются, если <выражение> = <значение1>
case {<значение_2>, <значение_3>, ... , <значение_k>}
        инструкции
       % выполняются, если <выражение> = <значение_i (i = 2:k)>
...
otherwise
        инструкции
        % выполняются, если <выражение> не совпало ни с одним из значений
end
```

Оператор switch ... case 1 ... case k ... otherwise ... end выполняет ветвления в зависимости от значений некоторой переменной или выражения.

Оператор переключения содержит:

- заголовок switch, за которым следует вычисляемое выражение (скаляр или строка);

- произвольное количество групп case. Заголовки групп состоят из слова case, за которым следует возможное значение выражения, расположенного на той же строке. Последующие строки содержат инструкции, которые выполняются для данного значения выражения. Выполнение продолжается до тех пор, пока не встретится следующий оператор case или оператор otherwise. На этом выполнение блока switch завершается;
- группу otherwise. Заголовок включает только слово otherwise; начиная со следующей строки размещаются инструкции, которые выполняются, если значение выражения оказалось не обработанным ни одной из групп case. Выполнение завершается оператором end. Оператор end является последним в блоке переключателя.

Оператор switch работает, сравнивая значение вычисленного выражения со значениями групп case. Для числовых выражений оператор case выполняется, если *<значение>==<выражение>*. Для строковых выражений оператор case *истинен*, если strcmp(*значение*, *выражение*) *истинно*.

Если возможные значения для группы case записаны в виде массива ячеек, то инструкции этой группы выполняются, если хотя бы один элемент массива ячеек совпадает с выражением заголовка switch. Если совпадения нет, то выполняется группа otherwise, если она существует.

Пример:

Рассмотрим оператор switch со следующими условиями: он проверяет переменную input_num; если input_num равно -1, 0 или 1, то операторы case выводят на экран соответствующее сообщения. Если значение выражения input_num не равно ни одному из этих значений, то выполнение переходит к оператору otherwise:

```
switch input_num
   case -1
       disp('минус один')
   case 0
       disp('нуль')
   case 1
       disp('плюс один')
   otherwise
       disp('другое значение')
end
```

Оператор switch может использовать множественное условие в единственной группе case посредством включения выражения case, если выражение для этого условия записано в виде массива ячеек:

```
switch var
   case 1
```

```
        disp('1')
    case{2,3,4}
        disp('2, или 3, или 4')
    case 5
        disp('5')
    otherwise
        disp('что-то другое')
end
```

Сопутствующие операторы: CASE, END, IF, OTHERWISE, WHILE.

TRY ... CATCH ... END **Перехват и обработка ошибок**

Синтаксис:

try *инструкции* catch *инструкции* end

Описание:

Блок try ... catch ... end функционирует следующим образом: в нормальном режиме выполняются только инструкции между операторами try и catch; в случае ошибки она перехватывается функцией lasterr и начинают выполняться инструкции между операторами catch и end. Если возникает ошибка при выполнении этих инструкций, то выполнение будет приостановлено, если только эта ошибка не будет перехвачена другим блоком try ... catch. При этом сообщение об ошибке может быть получено с помощью функции lasterr.

Сопутствующие операторы: EVAL, EVALIN, CATCH, END.

RETURN **Возврат в вызывающую функцию**

Синтаксис:

return

Описание:

Команда return выполняет нормальный возврат или в вызывающую функцию, или к режиму работы с клавиатурой. Эта команда позволяет также завершить режим работы с клавиатурой.

Пример:

Допустим, что некоторая М-функция determinant предназначена для вычисления определителя матрицы. Тогда в теле этой функции следовало бы учесть возможность входа с пустой матрицей. В этом случае выходному аргументу присваивается значение NaN и осуществляется выход из М-функции.

```
 function d = determinant(A)
if isempty(A)
  d = NaN;
```

```
  return
else
...
end
```

Сопутствующие операторы: BREAK, DISP, END, ERROR, FOR, IF, SWITCH, WHILE.

Передача аргументов М-функции

NARGCHK **Проверка количества входных аргументов**

Синтаксис:

msg = nargchk(low, high, number)

Описание:

Функция nargchk часто используется внутри М-файла, чтобы проверить правильность передачи аргументов.

Функция msg = nargchk(low, high, number) возвращает сообщение об ошибке, если количество входных аргументов number меньше, чем их возможное минимальное число low, или больше, чем возможное максимальное число high. Если количество входных аргументов находится в диапазоне между low и high, то результатом является пустой массив.

Пример:

Допустим, что задана некоторая М-функция foo:

function f = foo(x, y, z)

error(nargchk(2, 3, nargin))

Если обратиться к ней в форме

foo(1)

то появится следующее сообщение об ошибке:

Not enough input arguments.

Входных аргументов недостаточно.

Сопутствующие операторы: NARGIN, NARGOUT.

NARGIN, NARGOUT **Определение количества входных и выходных аргументов**

Синтаксис:

n = nargin

n = nargin(<*имя_функции*>)

n = nargout

n = nargout(<*имя_функции*>)

Описание:

Функции nargin и nargout позволяют определить количество входных и выходных аргументов функции в процессе ее выполнения.

Функции n = nargin(<*имя_функции*>) и n = nargout(<*имя_функции*>) возвращают соответственно количество входных и выходных аргументов М-функции с указанным именем. Результат может быть отрицательным, если количество входных или выходных аргументов переменно.

Пример:

В приведенном фрагменте процедуры fplot показано, как можно учесть необязательные входные и выходные переменные:

```
function [x0, y0] = fplot(fnamre, lims, npts, angl, subdiv)
% FPLOT Построение графика функции
% FPLOT(fnamre, lims, npts, angl, subdiv)
% Первые два аргумента являются обязательными;
% остальные - необязательные.
% FPLOT(...) строит график;
% [x, y] = FPLOT(...) возвращает массивы значений аргумента и функции
if nargin < 5, subdiv = 20; end
if nargin < 4, angl = 10; end
if nargin < 3, npts = 25; end
if nargout == 0
        plot(x, y)
else
        x0 = x;
        y0 = y;
end
```

Сопутствующие функции: VARARGIN, VARARGOUT, MFILENAME.

Переопределяемые методы:

```
help inline/nargin.m
help inline/nargout.m
```

VARARGIN **Список входных аргументов переменной длины**

Синтаксис:

```
z = function myfun(x, y, varargin)
```

Описание:

Переменная varargin позволяет объединить любое количество входных аргументов; она представляет собой массив ячеек, который содержит аргументы-опции вызываемой функции. Эта переменная должна быть последней в списке входов, а ее написание допускается только строчными буквами.

Пример:

Рассмотрим М-функцию

```
function myplot(x, varargin)
    plot(x, varargin{:})
```

Она объединяет все входные аргументы начиная со второго в одну переменную varargin. В свою очередь, при обращении к функции plot используется список значений varargin{:}, чтобы передать все задействованные аргументы. Например, при вызове функции myplot в форме

```
myplot(sin(0:.1:1), 'color', [.5 .7 .3], 'linestyle', ':');
```

переменная varargin - это массив ячеек размера 1×4, содержащий значения

```
'color', [.5 .7 .3], 'linestyle', ':'.
```

Сопутствующие функции: VARARGOUT, NARGIN, NARGOUT, INPUTNAME, FUNCTION, LISTS.

VARARGOUT **Список выходных аргументов переменной длины**

Синтаксис:

```
[z, varargout] = myfun(x, y, n)
```

Описание:

Переменная varargout позволяет объединить любое количество выходных аргументов; она представляет собой массив ячеек, который содержит аргументы-опции выхода функции. Эта переменная должна быть последней в списке выходов, а ее написание допускается только строчными буквами.

Переменная varargout не создается при вызове функции; ее необходимо создать при формировании выходов создаваемой М-функции, используя соответствующие операторы цикла, как это показано в нижеследующем примере.

Пример:

Рассмотрим М-функцию

```
function [s, varargout] = mysize(x)
    nout = max(nargout, 1)-1;
    s = size(x);
    for i=1:nout,
        varargout(i) = {s(i)};
    end
```

Здесь с использованием цикла объединены все выходные аргументы начиная со второго в одну переменную varargout.

Переменная varargout - это массив ячеек размера 1×nout, содержащий значения размерностей выходного многомерного массива. Например, при обращении вида

```
[s, rows, cols, pages] = mysize(rand(4, 5, 3));
```

возвращаются значения

```
s = [4 5 3], rows = 4, cols = 5, pages = 3.
```

Сопутствующие функции: VARARGIN, NARGIN, NARGOUT, FUNCTION, LISTS.

INPUTNAME **Определение имени переменной по номеру входного аргумента М-функции**

Синтаксис:

inputname(<*номер_входного_аргумента*>)

Описание:

Команда inputname(<*номер_входного_аргумента*>) возвращает имя переменной в рабочей области М-функции, которая соответствует входному аргументу с указанным номером. Если аргумент не имеет имени, например является выражением, то возвращается пустая строка (''). Эта команда может применяться только в теле М-функции.

Пример:

Рассмотрим М-функцию myfun.m:

```
function c = myfun(a, b)
   disp(sprintf('Первый аргумент " %s".', inputname(1)))
```

```
x = 5; y = 3; myfun(x, y)
Первый аргумент " x".
```

Обращение в форме

```
myfun(pi+1, pi-1)
```

выводит пустую строку

```
Первый аргумент " ".
```

Сопутствующие функции: NARGIN, NARGOUT, NARGCHK, MFILENAME.

Организация диалога с пользователем

Для обеспечения взаимодействия с пользователем в процессе выполнения М-файлов в системе MATLAB предназначены следующие команды:

input	Ввод информации пользователем
keyboard	Переключение на работу с клавиатуры из М-файла
pause	Приостановка выполнения М-файла
menu	Создание меню

INPUT **Ввод информации пользователем**

Синтаксис:

x = input('<*приглашение*>')

x = input('<*приглашение*>', 's')

Описание:

Функция x = input('<*приглашение*>') выводит на экран строку с приглашением и ожидает ввода выражения, допустимого в системе MATLAB: арифметическое выражение, имя встроенной функции или М-файла. Если функция имеет несколько выходных параметров, то выходной переменной x присваивается только первое значение.

Функция x = input('<*приглашение*>', 's') возвращает строку, которая введена пользователем. Если нажать клавишу Enter без ввода строки, то будет введен пустой массив.

Вводимая строка символов может содержать одну или несколько групп символов \n, которая соответствует переходу на следующую строку. Таким образом, строка приглашения может быть размещена на нескольких строках.

Для вывода обратного слэша \ нужно использовать комбинацию символов \\.

Пример:

Задействовать клавишу Enter для ввода ответа по умолчанию.

```
i = input('Продолжить расчет? Да/Нет [Да]: ', 's')
Продолжить расчет? Да/Нет [Да]:
i =     [ ]
if isempty(i), i = 'Да', end
i = Да
```

Сопутствующие функции и команды: KEYBOARD, GINPUT, UICONTROL, MENU.

KEYBOARD **Переключение на работу с клавиатуры из М-файла**

Синтаксис:

```
keyboard
```

Описание:

Команда keyboard, когда она включена в текст М-файла, прерывает его исполнение и передает управление клавиатуре. Этот специальный режим работы отмечается тем, что на экране появляется приглашение K>>. В этом режиме пользователь может проверить или изменить переменные, а также ему доступны все команды системы MATLAB. Работа в этом режиме будет завершена, если выполнить команду return; после этого управление будет передано М-файлу.

Режим работы с клавиатурой может быть использован для отладки М-файлов, хотя в современных версиях системы MATLAB присутствует специальный отладчик.

Сопутствующие команды: DBQUIT, DBSTOP, RETURN, INPUT.

PAUSE **Приостановка выполнения М-файла**

Синтаксис:

```
pause          pause (n)
pause on       pause off
```

Описание:

Команда pause приводит к паузе в работе системы и ожиданию нажатия любой клавиши для продолжения.

Команда pause(n) приостанавливает работу системы MATLAB на n секунд. Это позволяет привлечь внимание пользователя к выдаваемой на экран в процессе работы М-файла числовой или графической информации.

Команда pause on включает режим пауз.

Команда pause off выключает режим пауз. Это позволяет отладочные М-файлы и файлы-сценарии выполнять без остановов.

Важное значение имеет эта команда при выводе на экран терминала графиков. Если она не используется, то графики выводиться не будут.

Пример:

Если в приводимом ниже цикле команду pause не использовать, то промежуточные графики выводиться не будут; если использовать команду в форме pause <*без параметров*>, то потребуется подтверждать продолжение счета нажатием клавиши Enter; если указать длительность паузы n = 0, то этого вполне достаточно для беглого слежения за графиками.

```
for n = 3:22
    mesh(magic(n))
    pause(0)
end
```

Сопутствующие команды: drawnow.

MENU **Создание меню**

Синтаксис:

k = menu('<*заголовок*>', '*выбор* 1', '*выбор* 2', . . . '*выбор* n')

Описание:

Функция k = menu('<*заголовок*>', '*выбор* 1', '*выбор* 2', . . . '*выбор* n') при работе с оконным интерфейсом выводит на экран меню с заголовком и кнопками выбора. Выходному параметру k присваивается номер выбранной кнопки.

При вводе команды

k = menu('Выберите цвет','Красный','Зеленый','Синий')

высвечивается следующее меню:

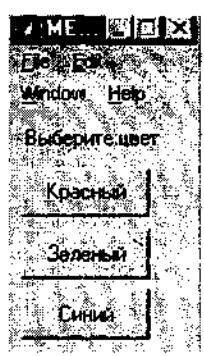


Рис. 5.2

При выборе синего цвета формируется ответ

```
k =    3
```

Теперь можно построить график функции s(t), окрашенный в выбранный цвет, в данном случае в синий:

```
color = ['r', 'g', 'b']
plot(t, s, color(k))
```

Сопутствующие функции и команды: UICONTROL, UIMENU.

Сообщения программы

ERROR — **Формирование сообщения об ошибке**

Синтаксис:

error('<*сообщение*>')

Описание:

Команда error('<*сообщение*>') используется для выдачи сообщения из М-файла при возникновении ошибки в процессе его выполнения или при прерывании работы М-файла по заданному условию.

Пример:

Команда error('<*сообщение*>') позволяет сформировать сообщение об ошибке при вызове или выполнении М-функции с именем foo:

```
function foo(x, y)
if nargin ~= 2
   error('Неверное количество входных параметров')
end
```

Сопутствующие функции и команды: DISP, DBSTOP, LASTERR, WARNING.

WARNING — **Формирование предупреждений**

Синтаксис:

warning('<*сообщение*>')	warning debug
warning off	warning once
warning on	warning always
warning backtrace	[s, f] = warning

Описание:

Команда warning('<*сообщение*>') используется для выдачи сообщения при возникновении предупреждения.

Команда warning off подавляет предупреждающие сообщения.

Команда warning on возобновляет выдачу предупреждающих сообщений после ее подавления.

Команда warning backtrace добавляет к предупреждающему сообщению имя файла и номер строки, где оно возникло.

Команда warning debug равносильна команде отладчика dbstop if warning и вызывает отладчик для работы в случае появления предупреждения.

Команда warning once выводит предупреждающее сообщение при работе подсистемы дескрипторной графики только один раз в течение сеанса.

Команда warning always выводит предупреждающее сообщение при работе подсистемы дескрипторной графики по мере их возникновения.

Функция [s, f] = warning возвращает в виде строки s текущую опцию команды warning: 'off', 'on', 'backtrace', 'debug' и частоту выдачи сообщений в виде строки f: 'once', 'always'.

Сопутствующие функции и команды: DISP, ERROR, DBSTOP.

LASTERR **Сообщение о последней ошибке**

Синтаксис:

lasterr

lasterr('<*пустая строка*>')

Описание:

Команда lasterr возвращает строку, содержащую сообщение о последней ошибке.

Команда lasterr('<*пустая строка*>') обнуляет сообщение об ошибке.

Обычно команда lasterr используется вместе с конструкциями EVAL('try', 'catch') и TRY...CATCH...END, в которых блоки обработки ошибок с ее помощью перехватывают сообщение об ошибке, анализируют его и выполняют соответствующее действие.

Пример:

М-функция catch(A, B) для обработки ошибок использует встроенную функцию lasterr, чтобы проверить сообщение об ошибке, которая возникает из-за несогласованности размеров операндов. Если возникает ошибка, то программа уменьшает размеры одной из матриц:

```
function C = catch(A, B)
l = lasterr;
j = findstr(l, 'Inner matrix dimensions');
if (~isempty(j))
  [m, n] = size(A)
  [p, q] = size(B)
  if (n>p)
    A(:, p+1:n) = []
  else if (n<p)
    B(n+1:p, :) = []
  end
  C = A*B;
else
   C = 0;
end
```

Использование функции eval с двумя аргументами, одним из которых является функция catch, показано ниже:

```
clear
A = [1 2 3; 6 7 2; 0 1 5];
B = [9 5 6; 0 4 9];
eval('A*B','catch(A, B)')
m =      3
n =      3
p =      2
q =      3
A =
     1     2
     6     7
     0     1
ans =
     9    13    24
    54    58    99
     0     4     9
```

Сопутствующие команды и функции: ERROR, EVAL, LASTWARN.

LASTWARN **Сообщение о последнем предупреждении**

Синтаксис:

lastwarn
lastwarn('*<пустая строка>*')
lastwarn('*<сообщение>*')

Описание:

Команда lastwarn возвращает строку, содержащую последнее предупреждающее сообщение, выданное системой MATLAB.

Команда lastwarn('*<пустая строка>*') обнуляет строку, в которой формируется предупреждающее сообщение.

Команда lastwarn('*<сообщение>*') заменяет последнее предупреждающее сообщение строкой '*<сообщение>*'. Это выполняется независимо от состояния опций on или off.

Сопутствующие команды и функции: LASTERR, WARNING.

ERRORTRAP **Флаг управления программой при возникновении ошибки**

Синтаксис:

errortrap off
errortrap on

Описание:

Команда errortrap off останавливает выполнение программы при возникновении ошибки; используется по умолчанию.

Команда errortrap on устанавливает флаг на продолжение выполнения программы при возникновении ошибки.

Сопутствующие команды и функции: HTML-справка.

DISP **Вывод значений переменных и текста на экран**

Синтаксис:

disp(<*переменная*> | '<*текст*>')

Описание:

Команда disp(X) выводит на терминал значение переменной X без указания ее имени.

Команда disp ('<*текст*>') выводит на терминал символьную строку '<*текст*>'.

После каждой команды disp происходит перевод на новую строку.

Пример:

```
disp('  Столбец 1  Столбец 2  Столбец 3' )
    disp(rand(5,3))
        Столбец 1  Столбец 2  Столбец 3
          0.4057     0.0579     0.2028
          0.9355     0.3529     0.1987
          0.9169     0.8132     0.6038
          0.4103     0.0099     0.2722
          0.8936     0.1389     0.1988
```

Сопутствующие команды: FORMAT, INT2STR, NUM2STR, RATS, SPRINTF.

Переопределяемые методы: help inline/disp.m.

FPRINTF, SPRINTF **Запись форматированных данных**

Синтаксис:

count = fprintf(<*идентификатор_файла*>, <*формат*>, A, ...)

fprintf(<*формат*>, A, ...)

s = sprintf(<*формат*>, A, ...)

[s, <*сообщение_об_ошибке*>] = sprintf(<*формат*>, A, ...)

Описание:

Функция count = fprintf(<*идентификатор_файла*>, <*формат*>, A, ...) преобразовывает данные в строки символов в соответствии с указанным форматом и выводит их на экран или в файл в зависимости от значения идентификатора файла. Идентификатор файла - это целое число, которое может быть получено с помощью функции fopen. Оно имеет значение 1 для вывода на экран и 2 в случае стандартной ошибки. По умолчанию идентификатор файла равен 1. В качестве выходного аргумента возвращается количество записанных байт.

Команда fprintf(<*формат*>, A, ...) выводит данные на экран терминала.

Функция s = sprintf(<*формат*>, A, ...) преобразовывает данные в строки символов в соответствии с указанным форматом и возвращает их в виде строковой переменной s, а не записывает их в файл.

Функция [s, <*сообщение_об_ошибке*>] = sprintf(<*формат*>, A, ...) возвращает, кроме того, сообщение об ошибке, если она имела место, либо пустую строку.

Особого внимания заслуживает обсуждение аргумента <*формат*>. Строка формата определяет способы записи, выравнивания, количество значащих цифр, ширину поля и другие характеристики описания формата данных. Она может содержать алфавитно-цифровые символы, символы перевода строки, спецификаторы преобразования и другие символы. Строка формата выглядит следующим образом:

%	+	12.5	e
начальный символ	флаг	ширина поля и точность	символ преобразования

Функции fprintf и sprintf аналогичны соответствующим функциям языка ANSI C за некоторыми исключениями:

- следующие нестандартные спецификаторы подтипа поддержаны для спецификаторов преобразования %o, %u, %x, %X:
  t - основной тип данных - числа с плавающей точкой, а не целое без знака;
  b - основной тип данных - числа с удвоенной точностью, а не целое без знака.
  Например, для вывода чисел с удвоенной точностью в шестнадцатеричной системе счисления следует применить формат %bx;
- функции fprintf и sprintf исполняются в векторном режиме, когда входом является массив. Строка формата циклически применяется к элементам столбца, пока не будут исчерпаны все элементы.

В следующей таблице представлены символы перевода строки.

Символ	*Описание*
\n	Перевод на новую строку
\t	Горизонтальная табуляция
\b	Возврат на один символ
\r	Возврат каретки
\f	Перевод страницы
\\	Наклонная черта влево
\" или " (две кавычки)	Одиночная кавычка
%%	Символ процента

Следующие спецификаторы преобразования определяют способ записи выходной строки.

Спецификатор	*Описание*
%c	Одиночный символ
%d	Десятичный формат со знаком
%e	Экспоненциальный формат вида 3.1415e + 00
%E	Экспоненциальный формат вида 3.1415E + 00
%f	Формат с фиксированной точкой
%g	Наиболее компактный из форматов %e и %f
%G	Наиболее компактный из форматов %E и %f
%o	Восьмеричная запись без знака
%s	Строка символов
%u	Десятичная запись без знака
%x	Шестнадцатеричная запись с символами a-f
%X	Шестнадцатеричная запись с символами A-F

Другие символы, которые могут быть использованы в строке формата:

Символ	*Описание*	*Пример*
-	Выравнять преобразуемое число по полю	%-5.2d
+	Всегда выводить знак числа	%+5.2d
0	Дополнять нулями, а не пробелами	%05.2d
Цифра (ширина поля)	Минимальное количество выводимых цифр числа	%6f
.цифра (точность)	Количество выводимых цифр после десятичной точки	%6.2f

Примеры:

Следующие операторы

```
x = 0 : .1 : 1;
y = [x; exp(x)];
fid = fopen('exp.txt', 'w');
fprintf(fid, '%6.2f %12.8f\n',y);
fclose(fid);
type exp.txt
```

создают текстовый файл exp.txt, который содержит таблицу экспоненты

```
 0.00  1.00000000
 0.10  1.10517092
 0.20  1.22140276
 0.30  1.34985881
 0.40  1.49182470
 0.50  1.64872127
 0.60  1.82211880
 0.70  2.01375271
 0.80  2.22554093
 0.90  2.45960311
 1.00  2.71828183
```

Команда

fprintf('Длина единичной окружности равна %g.\n',2*pi)

выводит на экран строку

Длина единичной окружности равна 6.28319.

Чтобы указать в строке одиночную кавычку, надо использовать две кавычки:

fprintf(1, 'It''s Friday.\n')

It's Friday.

Операторы

B = [8.8 7.7; 8800 7700];

fprintf(1,'X равно %6.2f м или %8.3f мм\n',9.9,9900,B)

выводят следующие строки:

X равно 9.90 м или 9900.000 мм

X равно 8.80 м или 8800.000 мм

X равно 7.70 м или 7700.000 мм

Преобразовать 32-разрядные числа со знаком в шестнадцатеричный формат:

a = [6 10 -10 14 44];

fprintf('%9X\n',a + (a<0)*2^32)

```
       6
       A
 FFFFFFF6
       E
      2C
```

Команда	*Результат*
sprintf('%0.5g',(1+sqrt(5))/2)	1.618
sprintf('%0.5g',1/eps)	4.5036e+15
sprintf('%15.5f',1/eps)	4503599627370496.00000
sprintf('%d',round(pi))	3
sprintf('%s','hello')	hello
sprintf('Размер массива %dx%d.',2,3)	Размер массива 2x3
sprintf('\n')	Символ конца строки

Сопутствующие команды: FSCANF, FWRITE, DIARY, INPUT, INT2STR, NUM2STR, SAVE, SSCANF.

Ссылки:

1. Kernighan B.W. and D.M. Ritchie, The C Programming Language, Second Edition, Prentice-Hall, Inc., 1988.
2. ANSI specification X3.159-1989: Programming Language C, ANSI, 1430 Broadway, New York, NY 10018.

Отладка и профилирование М-файлов

Отладка программного кода - это процесс, в ходе которого могут быть выявлены ошибки двух видов:

- *синтаксические*, которые связаны с неточностью записи имен М-функций или арифметических выражений. MATLAB обнаруживает большинство синтаксических ошибок, сопровождая их сообщением об ошибке с указанием номера строки соответствующего М-файла;
- *ошибки времени выполнения*, которые, как правило, имеют алгоритмическую природу и проявляются в том, что приводят к непредвиденным результатам.

Отладчик, реализованный в системе MATLAB, предназначен для выявления ошибок при программировании на языке MATLAB. С помощью отладчика можно просматривать состояние рабочей области в процессе выполнения, просматривать стек вызова М-функций, выполнять код М-файла построчно.

Отладчик может функционировать как в режиме командной строки, так и в режиме графического интерфейса пользователя.

Режим командной строки

DBSTOP **Установить контрольную точку**

Синтаксис:

dbstop [in] <*имя_М-функции*>
dbstop [in] <*имя_М-функции*> [at] <*номер_строки*>
dbstop [if] error
dbstop [if] naninf
dbstop [if] infnan

Описание:

Группа команд dbstop устанавливает режим отладки в среде системы MATLAB. Она позволяет установить контрольную точку в определенной строке М-функции или вызвать прерывание в случае возникновения предупреждения или ошибки. Если установленное условие оказалось выполненным, выводится специальное приглашение K>>, которое разрешает выполнить любую команду системы MATLAB.

Команда dbstop [in] <*имя М-функции*> останавливает исполнение процедуры в первой строке М-функции, как только она будет вызвана. Служебное слово in, помещенное в квадратные скобки [ ], является необязательным.

Команда dbstop [in] <*имя М-функции*> [at] <*номер строки*> останавливает выполнение в заданной строке указанного М-файла. В состав имени функции может быть включен путь доступа. Служебные слова in и at, помещенные в квадратные скобки [ ], являются необязательными.

Команда dbstop if error устанавливают контрольную точку по условию, связанному с возникновением ошибки при исполнении модуля. В случае возникновения такой ситуации можно проверить переменные рабочей области и последовательность вызова функций, приведших к ошибке. Однако продолжить выполнение М-файла оказывается невозможным.

Команды bstop if naninf и bstop if infnan устанавливают контрольную точку по условию, связанному с появлением результата NaN или inf.

Команда dbstop if warning устанавливают контрольную точку по условию, связанному с появлением предупреждения при исполнении модуля:

- если контрольная точка определяется номером строки, то исполнение модуля прерывается перед этой строкой;
- если именем модуля, то останов перед первой исполнимой строкой;
- если условием if error, то останов при появлении ошибки;
- если условиями if naninf, if infnan, то останов при получении результата NaN или inf.

Сопутствующие команды: DBCONT, DBSTEP, DBCLEAR, DBTYPE, DBSTACK, DBUP, DBDOWN, DBSTATUS, DBQUIT, PARTIALPATH.

DBCLEAR **Удалить контрольные точки**

Синтаксис:

dbclear in <*имя М-функции*>
dbclear in <*имя М-функции*> at <*номер строки*>
dbclear all in <*имя М-функции*>
dbclear all
dbclear if error
dbclear if warning
dbclear if naninf
dbclear if infnan

Описание:

Команды из группы dbclear удаляют контрольные точки, установленные ранее соответствующей командой dbstop.

Команда dbclear [in] <*имя М-функции*> удаляет все контрольные точки в данном М-файле. Служебное слово in, помещенное в квадратные скобки [ ], является необязательным.

Команда dbclear [in] <*имя М-функции*> [at] <*номер строки*> удаляет контрольную точку в заданной строке данной М-функции. Служебные слова in и at, помещенные в квадратные скобки [ ], являются необязательными.

Команда dbclear all [in] <*имя М-функции*> удаляет все контрольные точки в данном М-файле. Служебное слово in, помещенное в квадратные скобки [ ], является необязательным.

Команда dbclear all удаляет во всех активных М-функциях все контрольные точки, за исключением тех, которые связаны с фиксацией предупреждений и ошибок.

Команды dbclear if error и dbclear if warning удаляют контрольные точки, установленные соответственно командами dbstop if error и dbstop if warning.

Команды dbclear if naninf и dbclear if infnan удаляют контрольные точки, установленные соответственно командами dbstop if naninf и dbstop if infnan.

Сопутствующие команды: DBCONT, DBDOWN, DBQUIT, DBSTACK, DBSTATUS, DBSTEP, DBSTOP, DBTYPE, DBUP, PARTIALPATH.

DBSTEP — **Выполнить одну или несколько строк программы в режиме отладки**

Синтаксис:

dbstep
dbstep <*количество строк*>
dbstep in

Описание:

Группа команд dbstep позволяет управлять режимом отладки и включает 3 команды.

Команда dbstep реализует построчное исполнение М-функции.

Команда dbstep <*количество строк*> допускает исполнение сразу нескольких строк.

Команда dbstep in связана с исполнением строки, в которой присутствует вызов другой М-функции. В последнем случае, если следующая строка содержит вызов М-функции, применение команды dbstep in позволяет создать контрольную точку в первой строке вызываемой функции.

Сопутствующие команды: DBCLEAR, DBCONT, DBDOWN, DBQUIT, DBSTACK, DBSTATUS, DBSTOP, DBTYPE, DBUP.

DBCONT — **Продолжить выполнение**

Синтаксис:

dbcont

Описание:

Команда dbcont вызывает исполнение М-функции до следующей контрольной точки, установленной командами dbstop или dbstep.

Сопутствующие команды: DBCLEAR, DBDOWN, DBQUIT, DBSTACK, DBSTATUS, DBSTEP, DBSTOP, DBTYPE, DBUP.

DBSTACK **Стек вызываемых М-функций**

Синтаксис:

dbstack

[ST, I] = dbstack

Описание:

Команда dbstack выводит на терминал номера строк и имена вызванных М-функций, начиная от контрольной точки и до самого внешнего модуля, за исключением М-сценария (Script-файла).

Оператор [ST, I] = dbstack возвращает стек вызванных функций в виде массива записей (структуры) ST размера m×1 с полями ST.line, ST.name. Текущей рабочей области присваивается индекс I=1; при однократном использовании команды dbup индекс I увеличивается на 1, так что самый высокий индекс имеет базовая рабочая область системы MATLAB.

Пример:

Рассмотрим использование команды dbstack при останове в некоторой контрольной точке:

```
K» dbstack
> In d:\matlab5\sqsum.m at line 3
  In d:\matlab5\variance.m at line 3
```

Рассмотрим использование оператора [ST, I] = dbstack при останове в той же контрольной точке:

```
[ST, I] = dbstack
ST =
2x1 struct array with fields:
    line
    name
I =     1
```

Выведем содержимое полей ST.name и ST.line:

```
K» ST.name
ans = d:\matlab5\sqsum.m
ans = d:\matlab5\variance.m
K» ST.line
ans =     3
ans =     3
```

Сопутствующие команды: DBCLEAR, DBCONT, DBDOWN, DBQUIT, DBSTATUS, DBSTEP, DBSTOP, DBTYPE, DBUP.

DBUP **Переход между рабочими областями снизу вверх**

Синтаксис:

dbup

Описание:

Команда dbup осуществляет переход в стеке вызываемых М-функций снизу вверх. Все переменные доступны для просмотра (команды who, whos) и обработки, и можно проследить, как они изменялись вплоть до значения, которое было передано отлаживаемому модулю. Рабочая область переменных самого внешнего модуля называется *базовой рабочей областью*. Для продолжения отладки выполнения команды, обратной dbup (команда dbdown), не требуется.

Сопутствующие команды: DBCLEAR, DBCONT, DBDOWN, DBQUIT, DBSTACK, DBSTATUS, DBSTEP, DBSTOP, DBTYPE.

DBDOWN **Переход между рабочими областями сверху вниз**

Синтаксис:

dbdown

Описание:

Команда dbdown применяется совместно с командой dbup для перемещения между рабочими областями вызываемых модулей. Эта команда противоположна по своему действию команде dbup. Команда dbdown реализует перемещение только в том случае, если выполнена хотя бы одна команда dbup.

Сопутствующие команды: DBCLEAR, DBCONT, DBQUIT, DBSTACK, DBSTATUS, DBSTEP, DBSTOP, DBTYPE, DBUP.

DBSTATUS **Список контрольных точек данной М-функции**

Синтаксис:

dbstatus
dbstatus <*имя М-функции*>
s = dbstatus

Описание:

Команда dbstatus выводит на терминал список всех контрольных точек, включая контрольные точки, связанные с ошибками и предупреждениями, а также с результатами вычислений вида NaN и Inf.

Команда dbstatus <*имя М-функции*> выводит на терминал список всех контрольных точек, определенных для данной М-функции. Эту команду можно использовать в формах dbstatus class/<*имя М-функции*>, dbstatus private/<*имя М-функции*>, dbstatus private/class/<*имя М-функции*>, чтобы создать список контрольных точек соответственно для методов, частных функций и частных методов. Кроме того, во всех этих случаях можно связывать имя функции с подфункцией в форме dbstatus <*имя М-функции*>/<*имя подфункции*>.

Оператор s = dbstatus возвращает информацию о контрольных точках в виде массива записей (структуры) размера m×1 с полями s.name, s.line, s.cond, которые содержат имена М-функций, вектор номеров строк с контрольными точками, строки условий (error, warning или naninf).

Пример:

Получим список всех контрольных точек:

```
K» dbstatus
Breakpoint for d:\matlab5\sqsum.m is on line 3.
Breakpoints for d:\matlab5\variance.m are on lines 3, 4.
```

Получим список всех контрольных точек в виде массива записей размера 2x1:

```
K» s = dbstatus
s =
2x1 struct array with fields:
    name
    line
    cond
```

Выведем содержимое полей s.name, s.line, s.cond:

```
K» s.name                    K» s.line          K» s.cond
ans = d:\matlab5\sqsum.m     ans =   3          ans =   ''
ans = d:\matlab5\variance.m  ans =   3   4      ans =   ''
```

Получим список контрольных точек, когда в состав М-функции variance включена подфункция sqsum:

```
K» dbstatus
Breakpoint for d:\matlab5\variance.m is on line 4.
Breakpoint for d:\matlab5\variance.m (sqsum) is on line 7.
```

Получим список контрольных точек подфункции sqsum:

```
K»  dbstatus variance/sqsum
Breakpoint for d:\matlab5\variance.m (sqsum) is on line 7.
```

Сопутствующие команды: DBCLEAR, DBCONT, DBDOWN, DBQUIT, DBSTACK, DBSTEP, DBSTOP, DBTYPE, DBUP.

DBTYPE **Текст М-функции с указанием номеров строк**

Синтаксис:

dbtype *<имя М-функции>*

dbtype *<имя М-функции> <начало>* : *<конец>*

Описание:

Команда dbtype *<имя М-функции> <начало>* : *<конец>* позволяет вывести на терминал текст М-функции с указанием номеров строк; для вывода части текста следует указать диапазон номеров выводимых строк; для вывода одной строки достаточно указать ее номер.

Сопутствующие команды: DBCLEAR, DBCONT, DBDOWN, DBQUIT, DBSTACK, DBSTATUS, DBSTEP, DBSTOP, DBUP, PARTIALPATH.

DBQUIT **Выход из режима отладки**

Синтаксис:

dbquit

Описание:

Команда dbquit немедленно прекращает режим отладки и возвращает управление базисному модулю. Исполнение текущего М-файла прерывается, результаты не возвращаются. Все контрольные точки сохраняются. Если основной модуль является М-сценарием, то его выполнение также прерывается, появляется сообщение об ошибке и управление передается в среду системы MATLAB.

Пример:

Выход из режима отладки в случае, когда внешний модуль mslsnae2.m является М-сценарием (Script-файлом):

```
K» dbstack
In d:\toolbox\snae\pencil.m at line 20
In d:\toolbox\snae\msnae2.m at line 34
K» dbquit
Error in ==> d:\toolbox\snae2\mslsnae2.m
On line 70  ==>  [x, y, err] = msnae2(Axy, P, sx, sy, nv)
»
```

Сопутствующие команды: DBCLEAR, DBCONT, DBDOWN, DBSTACK, DBSTATUS, DBSTEP, DBSTOP, DBTYPE, DBUP.

Режим графического интерфейса

Рассмотрим процесс отладки с использованием отладчика M-File Editor/Debugger.

Для его запуска используется команда edit со следующим синтаксисом:

edit

edit <*имя_М-файла*>

edit <*имя_файла.****>

Команда edit запускает отладчик M-File Editor/Debugger.

Команда edit <*имя_М-файла*> запускает отладчик M-File Editor/Debugger и открывает М-файл с указанным именем.

Команда edit <*имя_файла.****> запускает отладчик M-File Editor/Debugger и открывает файл с расширением ***.

Экран отладчика приведен на рис. 5.3

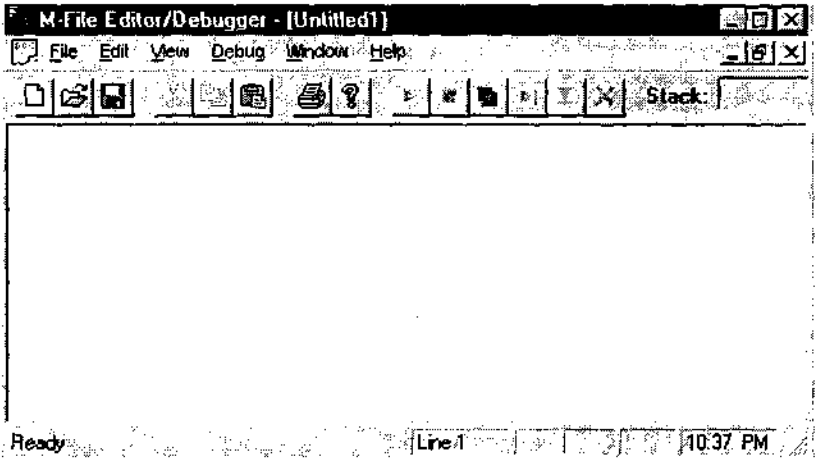


Рис. 5.3

Иконки отладки на инструментальной панели имеют следующее назначение:

Иконка отладки	*Описание*	*Команда*
Continue	Продолжить выполнение М-файла до его завершения или до первой контрольной точки	dbcont
Set/Clear Breakpoint	Установить/удалить контрольную точку в той строке, на которой находится курсор	dbstop/ dbclear
Clear All Breakpoints	Удалить все контрольные точки	dbclear all
Single Step	Исполнить текущую строку	dbstep
Step In	Исполнить текущую строку, а если она включает вызов М-функции, выполнить в ней останов	dbstep in
Quit Debugging	Выйти из режима отладки	dbquit

Профилировщик М-файлов

Один из способов повысить эффективность кодирования М-файлов - это профилировать их, то есть определить время, затрачиваемое на вычисление каждой строки.

Профилирование - это процедура измерения затрат времени на выполнение строк программы. Результат таких измерений часто оказывается неожиданным, поскольку затраты оказываются максимальными вовсе не там, где предполагалось. Поэтому первоначальную реализацию программы надо делать настолько простой, насколько это возможно, а затем использовать профилировщик, чтобы выявить критические участки программы, если быстродействие в самом деле является главным показателем эффективности создаваемой программы. Преждевременная оптимизация может усложнить код, не обеспечив реального повышения эффективности.

Профилировщик целесообразно использовать, чтобы выявить М-функции, которые требуют большого времени, затем определить, почему и как они вызываются, и поискать способы минимизации их использования. Часто бывает полезно задаться вопросом, а требуется ли вызывать М-функцию столько раз. Поскольку программы часто имеют несколько уровней, может оказаться, что созданный код вызывает наиболее трудоемкие функции неявно. Точнее говоря, функции внутри вашего кода могут вызывать другие функции, которые требуют большего времени и могут находиться на более низких уровнях. В этом случае важно определить, какие из функций высшего уровня являются ответственными за такие обращения.

Часто профилировщик помогает выявить проблемы, которые могут быть решены:

- отказом от лишних вычислений, которые могут быть следствием невнимательности;
- корректировкой алгоритма, чтобы избежать вызова неэффективных М-функций;
- отказом от многократных повторных вычислений путем хранения результатов для последующего использования.

Конечная цель профилирования состоит в том, чтобы повысить быстродействие программы. Как только достигается состояние, когда наибольшее время тратится на обращения к малому числу встроенных функций, то это означает, что достигнута оптимизация кода.

PROFILE **Измерить и вывести на экран профиль исполняемого М-файла**

Синтаксис:

```
profile <имя М-функции>
profile report _ | n | frac
profile plot
profile on | off | reset | done
info = profile
```

Описание:

Утилита профилировщика помогает отладить и оптимизировать М-функции, фиксируя время выполнения каждой строки программы. Утилита создает вектор измерений для каждой строки программы в профилируемом М-файле. При выполнении программы профилировщик обновляет вектор измерений с учетом времени, затрачиваемого на выполнение соответствующей строки.

Команда profile *<имя М-функции>* запускает профилировщик для заданной функции, имя которой должно быть именем М-файла, возможно с указанием частичного пути доступа.

Команда profile report _ | n | frac выводит на экран либо полный отчет о профиле М-файла (в отсутствие каких-либо опций), либо только об n строках

с наибольшим временем исполнения, либо о тех строках, доля затраченного времени для которых от общего времени выполнения превышает значение frac из диапазона от 0 до 1.

Команда profile plot выводит на экран результаты профилирования в виде диаграммы Парето.

Команды profile on и profile off соответственно запускают или приостанавливают процесс профилирования; команда profile reset очищает векторы измерений, не отключая профилировщик; команда profile done завершает работу профилировщика и удаляет сопутствующие данные.

Оператор info = profile возвращает результаты профилирования в виде структуры со следующими полями:

file	Полный путь доступа к профилируемой функции
function	Имя профилируемой функции
interval	Интервал измерения в секундах
count	Вектор измерений
state	Состояние профилировщика: on - активен; off - не активен

Профилировщик отслеживает количество интервалов, затраченных на выполнение встроенной функции.

Поведение профилировщика зависит от свойств корневого объекта и может управляться с помощью команд set и get.

Ограничение:

Одновременно профилировщик может обрабатывать только один М-файл.

Пример:

```
profile ellipke
[k, e] = ellipke(.01:.01:.99);
profile report
   Total time in
   "d:\matlab5\toolbox\matlab\specfun\ellipke.m": 0.19
   seconds
   Полное время исполнения модуля
   "d:\matlab5\toolbox\matlab\specfun\ellipke.m": 0.19 с

   100% of the total time was spent on lines:
   100% времени затрачено на выполнение строк:
    [39 47 55 52 50 48 45 44 43 34]

                  33: if isempty(m), k = zeros(size(m)); e = k;
   return, end
   0.01s,  5%     34: if any(m(:) < 0) | any(m(:) > 1),
                  35:   error('M must be in the range 0<M<1.');
```

```
                    38: a0 = 1;
0.06s, 32%          39: b0 = sqrt(1-m);
                    40: s0 = m;

                    42: while mm > tol
0.01s,  5%          43:     a1 = (a0+b0)/2;
0.01s,  5%          44:     b1 = sqrt(a0.*b0);
0.01s,  5%          45:     c1 = (a0-b0)/2;
                    46:     i1 = i1 + 1;
0.05s, 26%          47:     w1 = 2^i1*c1.^2;
0.01s,  5%          48:     mm = max(w1(:));
                    49:     s0 = s0 + w1;
0.01s,  5%          50:     a0 = a1;
                    51:     b0 = b1;
0.01s,  5%          52: end
                    53: k = pi./(2*a1);
                    54: e = k.*(1-s0/2);
0.01s,  5%          55: im = find(m ==1);
                    56: if ~isempty(im)
EDU» profile plot
```

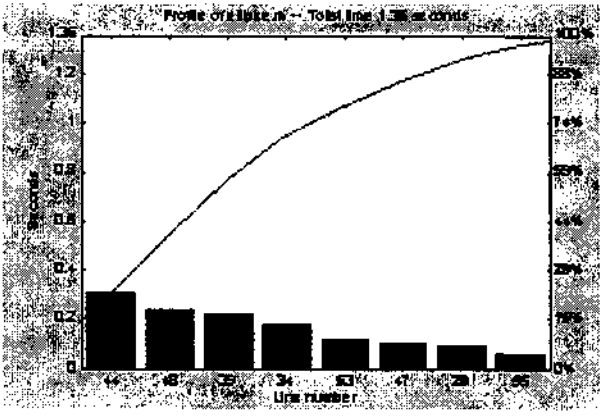


Рис. 5.4

Сопутствующие команды: DEBUG, PROFSUMM.

PROFSUMM **Сформировать отчет о профиле**

Синтаксис:

profsumm	profsumm(info)
profsumm(fraction)	profsumm(info, fraction)
profsumm(n)	profsumm(info, n)
profsumm(str)	profsumm(info,str)

Описание:

Команда profsumm выводит на экран полный отчет о профиле исследуемогоМ-файла.

Команда profsumm(fraction), где fraction - значение от 0 до 1, выводит на экран часть отчета, которая включает только те строки, время выполнения которых составляет не менее чем долю fraction от полного времени.

Команда profsumm(n), где n - положительное число, выводит на экран часть отчета, которая включает только n строк, время выполнения которых максимально.

Команда profsumm(str), где str - строковое выражение, выводит на экран результаты выполнения только тех строк, которые включают заданное строковое выражение.

Команды profsumm(info), profsumm(info, fraction), profsumm(info, n), profsumm(info, str) выводят на экран итоговый отчет, используя информацию из массива записей info.

Пример:

Сформируем итоговый отчет, который включает только те строки, время исполнения которых составляет не менее 0.1 от полного времени выполнения модуля ellipke.m:

```
profsumm(0.1)
Total time in "d:\matlab5\toolbox\matlab\specfun\ellipke.m":
0.31 seconds
65% of the total time was spent on lines:
          [34 39 47]
            33: if isempty(m), k = zeros(size(m)); e = k;
                   return, end
0.04s, 13%  34: if any(m(:) < 0) | any(m(:) > 1),
            35:   error('M must be in the range 0<M<1.');

            38:     a0 = 1;
0.06s, 19%  39:     b0 = sqrt(1-m);
            40:     s0 = m;

            46:     i1 = i1 + 1;
0.10s, 32%  47:     w1 = 2^i1*c1.^2;
            48:     mm = max(w1(:));
```

Сопутствующие команды: DEBUG, PROFILE.

PARETO **Построение диаграммы Парето**

Синтаксис:

pareto(y, names)
pareto(y, x)
pareto(y)
[h, ax] = pareto(...)

Описание:

Команда pareto(y, names) строит диаграмму Парето, на которой значения вектора y изображаются в виде столбцов в порядке убывания значений y. Каждый столбец может быть помечен некоторым именем, указанным в массиве строк или массиве ячеек names.

Команда pareto(y, x) строит диаграмму Парето для значений вектора y в зависимости от значений вектора x.

Команда pareto(y) строит диаграмму Парето для значений вектора y в зависимости от его индексов.

Оператор [h, ax] = pareto(...) возвращает вектор графических поддержек h для линий и многоугольников диаграммы Парето, а также поддержки для графических осей в векторе ax.

Пример:

Функция pareto позволяет достаточно просто реализовать графический образ результатов профилирования.

```
profile erfcore
z = erf(0:.01:100);
profile report
Total time in "d:\matlab5\toolbox\matlab\specfun\erfcore.m":
17.76 seconds

79% of the total time was spent on lines:
   [99 91 25 20 94 48 97 87 109 100]

              19:     end
1.62s,   9%   20:     result = repmat(NaN,size(x));
              21: %

              24:     xbreak = 0.46875;
1.63s,   9%   25:     k = find(abs(x) <= xbreak);
              26:     if ~isempty(k)

              47: %
1.30s,   7%   48:    k = find((abs(x)>xbreak) & (abs(x)<=4.));
              49:     if ~isempty(k)

              86:     y = abs(x(k));
0.62s,   3%   87:     z = 1 ./ (y .* y);
              88:     xnum = p(6).*z;

              90:     for i = 1:4
1.86s,  10%   91:        xnum = (xnum + p(i)) .* z;
              92:        xden = (xden + q(i)) .* z;
              93:     end
1.46s,   8%   94:    result(k) = z.*(xnum + p(5))./(xden+q(5));
              95:    result(k) = (1/sqrt(pi) - result(k))./y;
```

```
                 96:     if jint ~= 2
0.88s,   5%      97:     z = fix(y*16)/16;
                 98:   del = (y-z).*(y+z);
3.63s,  20%      99: result(k) = exp(-z.*z).* exp(-del).*result(k)
0.50s,   3%     100:     k = find(~isfinite(result));
                101:                 result(k) = 0*k;

                108:             k = find(x > xbreak);
0.53s,   3%     109: result(k) = (0.5 - result(k)) + 0.5;
                110:             k = find(x < -xbreak);
```

Теперь сформируем выход профилировщика, используя команду profile без входных аргументов:

```
t = profile
t =
        file:
'd:\matlab5\toolbox\matlab\specfun\erfcore.m'
    interval: 0.0100
       count: [121x1  double]
       state: 'off'
```

Выход t - это структура, которая содержит результаты профилирования функции erfcore.m в поле count. Чтобы увидеть эти результаты, надо воспользоваться функцией pareto в форме:

```
pareto(t.count)
```

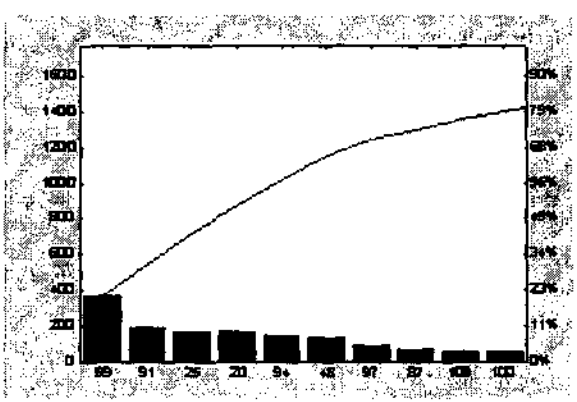


Рис. 5.5

В нижней части графика указаны номера строк по мере убывания времени их исполнения; на левой вертикальной оси отмечено время исполнения в интервалах по 0.01 с; на правой вертикальной оси отмечены проценты от полного времени; сплошная линия отражает суммарное время исполнения.

Сопутствующие команды: HIST, BAR.

6. МАТЕМАТИЧЕСКИЕ ФУНКЦИИ

В системе MATLAB имеется обширная библиотека математических функций. Каждой функции соответствует определенное имя. Функция ставит в соответствие значениям своих аргументов значение результата.

Аргументы функции всегда указываются в круглых скобках после имени функции и, если их больше одного, разделяются запятыми. В качестве аргументов могут использоваться другие функции и любые выражения языка MATLAB (при условии соответствия типов аргументов).

Элементарные функции

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

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

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

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

Функции обработки комплексных чисел

ABS **Абсолютное значение**

Синтаксис:

Y = abs(X)

Описание:

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

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

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

Примеры:

```
abs(-5) = 5
abs(3 + 4i) = 5

ascii = abs('3 + 4I')
ascii = 51  32  43  32  52  105

setstr(ascii)
ans = 3 + 4I
```

Сопутствующие функции: SIGN, ANGLE.

ANGLE **Аргумент комплексного числа**

Синтаксис:

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

Описание:

Для массивов комплексных чисел Z функция P = angle(Z) возвращает массив значений аргументов для элементов Z. Значение аргумента измеряется в радианах и находится в пределах от $-\pi$ до π.

Пример:

Для комплексного числа $z = x + iy = re^{i\varphi}$ его модуль r и аргумент φ вычисляются следующим образом:

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

а оператор

```
z = r .*exp(i*phi)
```

выполняет обратное преобразование.

Алгоритм:

Для вычисления аргумента комплексного числа используется следующее соотношение:

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

Сопутствующие функции: ABS, REAL, IMAG.

REAL, IMAG **Действительная и мнимая части комплексного числа**

Синтаксис:

```
X = real(Z)
Y = real(Z)
```

Описание:

Для массивов комплексных чисел Z функция X = real(Z) возвращает массив действительных, а Y = imag(Z) - мнимых частей элементов Z.

Сопутствующие функции: ABS, ANGLE, CONJ.

CONJ **Операция комплексного сопряжения**

Синтаксис:

V = conj(Z)

Описание:

Для массивов комплексных чисел Z функция V = conj(Z) возвращает массив комплексно-сопряженных значений для элементов Z.

Сопутствующие функции: IMAG, REAL.

CPLXPAIR **Сортировка комплексно-сопряженных пар**

Синтаксис:

V = cplxpair(Z)
V = cplxpair(Z, tol)
V = cplxpair(Z, [ ], dim)
V = cplxpair(Z, tol, dim)

Описание:

Для векторов комплексных чисел Z функция V = cplxpair(Z) сортирует эти числа в порядке возрастания действительных частей; в случае комплексно-сопряженных пар первым идет элемент с отрицательной мнимой частью; действительные элементы располагаются вслед за комплексными. Комплексно-сопряженные значения принудительно округляются в пределах погрешности 100*eps относительно abs(A(i)). Это гарантирует в дальнейшем точное совпадение действительных и модулей комплексных частей.

Для двумерных массивов комплексных чисел Z функция V = cplxpair(Z) выполняет сортировку по столбцам.

Для многомерных массивов комплексных чисел Z функция V = cplxpair(Z) выполняет сортировку начиная с первой размерности, размер которой отличен от 1, возвращая массив отсортированных элементов.

Функция V = cplxpair(Z, tol) позволяет переопределить принятую по умолчанию погрешность округления tol.

Функция V = cplxpair(Z, [ ], dim) выполняет сортировку вдоль размерности dim.

Функция V = cplxpair(Z, tol, dim) выполняет сортировку с заданной погрешностью округления tol и вдоль указанной размерности dim.

Диагностика:

Если количество сортируемых комплексных чисел нечетно или если невозможно сгруппировать пары в пределах указанной точности, формируется сообщение об ошибке:

Complex numbers can't be paired.

Комплексные числа не могут быть отсортированы попарно.

Пример:

Рассмотрим матрицы A, A^2, A^3 с двумя кратными комплексными и одним действительным собственным значением в виде четырехмерного массива, вдоль четвертой размерности которого размещаются соответствующие собственные значения eig(A), eig(A^2), eig(A^3):

```
A = [15   11   6   -9   -15
      1    3   9   -3    -8
      7    6   6   -3   -11
      7    7   5   -3   -11
     17   12   5  -10   -16]
```

Сформируем четырехмерный массив B, разместив в нем матрицы A, A^2, A^3 и их собственные значения:

```
B(:, :, 1, 1) = A;
B(:, :, 2, 1) = A^2;
B(:, :, 3, 1) = A^3;
B(:, 1, 1, 2) = eig(A);
B(:, 1, 2, 2) = eig(A^2);
B(:, 1, 3, 2) = eig(A^3)
B(:, :, 1, 1) =
   15   11    6   -9   -15
    1    3    9   -3    -8
    7    6    6   -3   -11
    7    7    5   -3   -11
   17   12    5  -10   -16

B(:, :, 2, 1) =
  -40   -9  105   -9   -40
  -76  -43   32   44    23
  -55  -22   62   20   -10
  -61  -25   65   20    -7
  -40   -9  110  -14   -40

B(:, :, 3, 1) =
 -617 -380   64  499   256
 -260 -189 -316  355   280
 -443 -279 -106  415   259
 -464 -300 -136  439   292
 -617 -385   69  499   256

B(:, :, 1, 2) =
     -1.0000e+000                0                0                0                0
 1.5000e+000 +3.5707e+000i       0                0                0                0
 1.5000e+000 -3.5707e+000i       0                0                0                0
 1.5000e+000 +3.5707e+000i       0                0                0                0
 1.5000e+000 -3.5707e+000i       0                0                0                0
```

```
B(:, :, 2, 2) =
-1.0500e+001+1.0712e+001i    0    0    0    0
-1.0500e+001 -1.0712e+001i   0    0    0    0
-1.0500e+001 +1.0712e+001i   0    0    0    0
-1.0500e+001 -1.0712e+001i   0    0    0    0
     1.0000e+000             0    0    0    0

B(:, :, 3, 2) =
    -1.0000e+000             0    0    0    0
-5.4000e+001 +2.1424e+001i   0    0    0    0
-5.4000e+001 -2.1424e+001i   0    0    0    0
-5.4000e+001 +2.1424e+001i   0    0    0    0
-5.4000e+001 -2.1424e+001i   0    0    0    0
```

Выполним сортировку комплексных значений с точностью, принятой по умолчанию:

```
D = cplxpair(B, [ ], 3);
??? Error using ==> cplxpair
Complex numbers can't be paired.
```

В силу кратности комплексных значений выдается сообщение о невозможности их сортировки по парам. Понизим точность сортировки до величины 1e0, что соответствует сортировке только действительных частей:

```
D = cplxpair(B, 1e0, 3);  D(:, 1, :, 2)
ans(:, :, 1) =          ans(:, :, 2) =          ans(:, :, 3) =
 -1.0500e+001            -1.0000e+000            -1.0000e+000
 -5.4000e+001            -1.0500e+001             1.5000e+000
 -5.4000e+001            -1.0500e+001             1.5000e+000
 -5.4000e+001            -1.0500e+001             1.5000e+000
 -5.4000e+001             1.0000e+000             1.5000e+000
```

Сопутствующие функции: IMAG, REAL.

Округление и модульная арифметика

CEIL, FIX, FLOOR, ROUND **Функции округления**

Синтаксис:

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

Описание:

Для массивов действительных чисел X:

- функция Y = ceil(X) возвращает значения, округленные до ближайшего целого ≥ X;

- функция Y = fix(X) возвращает значения с усечением дробной части числа;
- функция Y = floor(X) возвращает значения, округленные до ближайшего целого ≤ X;
- функция Y = round(X) возвращает значения, округленные до ближайшего целого.

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

Пример:

Задан одномерный массив действительных чисел

x = [-1.9 -0.2 3.4 5.6 7.0]

ceil(x)	[-1 0 4 6 7]	fix(x)	[-1 0 3 5 7]
floor(x)	[-2 -1 3 5 7]	roundl(x)	[-2 0 3 6 7]

Сопутствующие функции: CEIL, FIX, FLOOR, ROUND.

MOD, REM **Функции остатка**

Синтаксис:

M = mod(X,Y)
M = rem(X,Y)

Описание:

В случае скалярных аргументов функция m = mod(x, y) в полной мере соответствует математической функции модульной арифметики a mod b и вычисляет остаток от деления числа x на число y, то есть mod(x, y) = x - y*floor(x/y), если y не равно нулю. По определению mod(x, 0) = x.

В случае скалярных аргументов функция m = rem(x, y) вычисляет остаток от деления числа x на число y согласно соотношению rem(x, y) = x - y*fix(x/y), если y не равно нулю; rem(x, 0) = NaN.

Когда операнды одного знака, функция mod(x, y) равносильна функции rem(x, y). Однако для положительных x и y, когда знак x изменяется, верно следующее соотношение: mod(-x, y) = rem(-x, y) + y.

Функция mod(x, y) очень полезна для установления свойства конгруентности двух чисел. Два числа называются *конгруентными по модулю* m, если выполняется условие mod(x, m) = mod(y, m).

Для массивов чисел эти функции применяются поэлементно.

Пример:

```
mod(-13, 5)
ans = 2
```

```
mod(-[1:5], 3)
ans = 2 1 0 2 1
```

```
mod(-magic(3), 3)
ans =
   1   2   0
   0   1   2
   2   0   1
```

```
rem(-13, 5)
ans = -3
```

```
rem(-[1:5], 3)
ans = -1 -2  0  -1  -2
```

```
rem(-magic(3), 3)
ans =
   -2   -1    0
    0   -2   -1
   -1    0   -2
```

Сопутствующие функции: HTML-справка.

SIGN **Вычисление знака числа**

Синтаксис:

S = sign(Z)

Описание:

Для массивов действительных чисел X функция S = sign(X) возвращает массив S тех же размеров, в котором на месте положительного числа стоит 1, на месте нулевого - 0, на месте отрицательного - (-1).

Для массивов комплексных чисел Z функция S = sign(Z) возвращает массив комплексных чисел S = Z ./abs(Z), модуль которых равен единице.

Сопутствующие функции: ABS, IMAG, REAL.

Теоретико-числовые функции

FACTOR **Разложение числа на простые множители**

Синтаксис:

f = factor(n)

Описание:

Функция f = factor(n) возвращает все простые множители числа n.

Для массивов чисел эту функцию применять нельзя.

Пример:

```
factor(2^8 - 1)
ans = 3    5    17
```

Сопутствующие функции: ISPRIME, PRIMES.

PRIMES **Найти простые числа**

Синтаксис:

p = primes(n)

Описание:

Функция p = primes(n) возвращает все простые числа в диапазоне от 0 до n.

Пример:

p = primes(37)
p = 2 3 5 7 11 13 17 19 23 29 31 37

Сопутствующие функции: FACTOR, ISPRIME.

GCD **Наибольший общий делитель**

Синтаксис:

g = gcd(m, n)
[g, c, d] = gcd(m, n)
G = gcd(A, B)
[G, C, D] = gcd(A, B)

Описание:

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

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

Функция G = gcd(A, B) возвращает массив G, содержащий наибольшие общие делители целочисленных массивов A и B.

Функция [G, C, D] = gcd(A, B) возвращает массив наибольших общих делителей G и массивы C и D, удовлетворяющие следующему уравнению: A(i).*C(i) + B(i).*D(i) = G(i). Эта функция оказывается весьма полезной при решении диофантовых уравнений и вычислении преобразований Эрмита.

Примеры:

Первый пример связан с преобразованиями Эрмита. Для любых двух целых чисел m и n найдется такая целочисленная матрица E, определитель которой равен 1 и которая удовлетворяет уравнению E * [m; n] = [g, 0], где g - наибольший общий делитель чисел m и n, вычисляемый с помощью функции [g, c, d] = gcd(m, n). Матрица E при этом равна

$$\mathbf{E}=\begin{bmatrix} c & d \\ -n/g & m/g \end{bmatrix}.$$

Если m = 2 и n = 4, то
[g, c, d] = gcd(2, 4)
g = 2
c = 1
d = 0
и E равно
E =
1 0
-2 1

В следующем примере решим диофантово уравнение 30x + 56y = 8.
[g, c, d] = gcd(30, 56)

g = 2
c = –13
d = 7

По определению для скаляров c и d справедливо соотношение

$$30(-13) + 56(7) = 2,$$

умножим его на 8/2:

$$30(-13*4) + 56(7*4) = 8.$$

Сравнивая с исходным уравнением, получим решение:

$$x = (-13*4) = -52;\ y = (7*4) = 28.$$

Сопутствующие функции: ABS, FLOOR, ROUND, SIGN.

Ссылки:

1. Knuth D. *The Art of Computer Programming*. Vol. 2. Addison-Wesley: Reading MA, 1973. Section 4.5.2, Algorithm X.

LCM — Наименьшее общее кратное

Синтаксис:

```
g = lcm(m, n)
L = lcm(A, B)
```

Описание:

Функция g = lcm(m, n) вычисляет наименьшее общее кратное двух целых чисел m и n.

Функция L = lcm(A, B) вычисляет наименьшее общее кратное двух целых чисел m и n.

Функция L = lcm(A, B) возвращает массив L, содержащий наименьшие общие кратные для соответствующих пар элементов целочисленных массивов A и B.

Пример:

```
g = lcm(45, 36)
g = 180
lcm(pascal(3), magic(3))
ans =
   8   1   6
   3  10  21
   4   9   6
```

Сопутствующие функции: GCD.

RAT, RATS — Представление результата в виде рационального числа или цепной дроби

Синтаксис:

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

Описание:

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

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

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

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

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

Функция S = rats(X, k) использует функцию rat(X), чтобы вывести на экран результат в виде простой дроби

s = [sprintf(['%' num2str(fix(k/2)), n) '/' sprintf(['%-' num2str(fix(k/2)) '.0f'], d)],

точность аппроксимации для которой составляет tol = 10^(-fix(k/2)) * abs(x).

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

Функция format rat равносильна функции rats.

Алгоритм:

Функция rat(X) аппроксимирует каждый элемент массива X цепной дробью следующего вида:

$$\frac{n}{d} = d_1 + \cfrac{1}{d_2 + \cfrac{1}{(d_3 + \ldots \frac{1}{d_k})}}$$

Величины dj, j=1:k получены последовательным выделением целой части с последующим обращением дробной части. Точность аппроксимации возрастает по степенному закону с ростом числа членов. Самая медленная сходимость наблюдается при рациональной аппроксимации числа x = sqrt(2). Погрешность аппроксимации с учетом k членов составляет 2.68 * (0.173)^k, так что учет каждого последующего члена увеличивает точность менее чем на одну десятичную цифру, так что для достижения максимальной точности в арифметике с плавающей точкой требуется 21 член.

Примеры:

Рассмотрим аппроксимацию числа π в виде цепной дроби и рационального числа

```
rat(pi)
3 + 1/(7 + 1/(16))
rat(pi, 1e-12)
3 + 1/(7 + 1/(16 + 1/(-294 + 1/(3 + 1/(-4 + 1/(5))))))

[n, d] = rat(pi); [n d]
ans =    355   113

[n, d] = rat(pi, 1e-12); [n d]
ans =     5419351     1725033

s = rats(pi)
s =    355/113

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

Сопутствующие функции: FORMAT RAT.

PERMS **Формирование всех перестановок элементов вектора**

Синтаксис:

```
P = perms(v)
```

Описание:

Функция P = perms(v), где v - вектор-строка длины n, формирует массив, состоящий из всех возможных перестановок элементов этого вектора. Массив P имеет размер n!×n.

Пример:

Вычислим все возможные перестановки трех чисел 2, 4, 6:

```
  perms(2:2:6)
ans =
     6     4     2
     4     6     2
     6     2     4
     2     6     4
     4     2     6
     2     4     6
```

Ограничение:

Практическое применение этой функции ограничено значением n, меньшим 15.

Сопутствующие функции: NCHOOSEK, PERMUTE, RANDPERM.

NCHOOSEK **Вычисление числа сочетаний C_n^k и формирование всех сочетаний элементов вектора из n по k**

Синтаксис:

```
C = nchoosek(n, k)
C = nchoosek(v, k)
```

Описание:

Функция C = nchoosek(n, k), где n и k - неотрицательные целые числа, вычисляет число сочетаний C_n^k = n! / ((n–k)! k!).

Функция C = nchoosek(v, k), где v - вектор-строка длины n, формирует массив, состоящий из всех возможных сочетаний по k элементов этого вектора. Массив C имеет размер n! / ((n–k)! k!) × n.

Пример:

Вычислим все возможные сочетания из четырех чисел 2, 4, 6, 8 по 3:

```
nchoosek(2:2:8, 3)
ans =
    2     4     6
    2     4     8
    2     6     8
    4     6     8
nchoosek(4, 3)
ans =   4
```

Ограничение:

Практическое применение этой функции ограничено значением n, меньшим 15.

Сопутствующие функции: PERMS.

Трансцендентные функции

SQRT **Квадратный корень**

Синтаксис:

V = sqrt(Z)

Описание:

Функция V = sqrt(Z) вычисляет квадратные корни элементов массива Z.

Для отрицательных и комплексных значений результат является комплексным числом.

Пример:

```
w = sqrt((-2:2)')
w =
      0 + 1.4142i
      0 + 1.0000i
      0
   1.0000
   1.4142
```

Сопутствующие функции: EXP, LOG, SQRTM.

EXP **Экспоненциальная функция**

Синтаксис:

V = exp(Z)

Описание:

Функция V = exp(Z) вычисляет экспоненты значений элементов массива Z. Для комплексных значений z = x + iy справедлива формула Эйлера

$e^z = e^x (\cos(y) + i \sin(y))$.

Вычисление экспоненты от матрицы реализовано с помощью специальной функции expm.

Сопутствующие функции: LOG, LOG2, LOG10, EXPM.

LOG **Функция натурального логарифма**

Синтаксис:

V = log(Z)

Описание:

Функция V = log(Z) вычисляет натуральный логарифм значений элементов массива Z. Для комплексных значений z = x + iy справедлива формула

ln(z) = ln(abs(z)) + i atan2(y, x).

Вычисление функции натурального логарифма от матрицы реализовано с помощью специальной функции logm.

Пример:

Одна из возможностей вычисления значения числа π - это вычислить log(-1):

```
log(-1)
ans =
        0 +3.141592653589793e+000i
```

Сопутствующие функции: EXP, LOG2, LOG10, LOGM.

POW2 **Экспонента по основанию 2**

Синтаксис:

V = pow2(Z) X = pow2([M, P])

Описание:

Функция V = pow2(Z) вычисляет массив степеней 2.^Z.

Функция X = pow2([M, P]) для действительных массивов M и P вычисляет массив X = M.*(2.^P).

Пример:

Для компьютеров с IEEE-арифметикой, в которых определены объекты eps, realmax и realmin, функция x = pow2([m, p]) вычисляет следующие величины:

m	p	x
1/2	1	1
pi/4	2	pi
-3/4	2	-3
1/2	-51	eps
1-eps/2	1024	realmax
1/2	-1021	realmin

Сопутствующие функции: LOG2, NEXTPOW2.

NEXTPOW2 **Ближайшая степень по основанию 2**

Синтаксис:

p = nextpow2(n)
p = nextpow2(x)

Описание:

Функция p = nextpow2(n) возвращает такой показатель степени p, что 2^p ≥ n.

Функция p = nextpow2(x) для одномерного массива x возвращает значение nextpow2(length(x)). Эта операция широко применяется при вычислении быстрого преобразования Фурье.

Пример:

Для любого целого числа n в диапазоне от 513 до 1024 функция nextpow2(n) возвращает значение 10.

Сопутствующие функции: FFT, LOG2, POW2.

LOG2 **Функции логарифма**

Синтаксис:

V = log2(Z) [M, P] = log2(X)

Описание:

Функция V = log2(Z) вычисляет логарифм по основанию 2 от значений элементов массива Z.

Функция [M, P] = log2(X) для массива X действительных чисел возвращает массив M значений мантисс и целочисленный массив P показателей степеней, позволяющих представить любой элемент x в виде x = f*2^p; нулевому элементу соответствует представление {f = 0, e = 0}.

Примеры:

Для компьютеров с IEEE-арифметикой, в которых определены объекты eps, realmax, realmin, функция log2 вычисляет следующие величины:

log2(eps) = -52, log2(realmax) = 1024, log2(realmin) = -1022,

а функция [M, P] = log2(X) строит следующие представления чисел:

x	m	p
1	1/2	1
pi	pi/4	2
-3	-3/4	2
eps	1/2	-51
realmax	1-eps/2	1024
realmin	1/2	-1021

Сопутствующие функции: LOG2, NEXTPOW2, POW2.

LOG10 **Функции логарифма**

Синтаксис:

V = log10(Z)

Описание:

Функция V = log10(Z) вычисляет логарифм по основанию 10 от значений элементов массива Z.

Примеры:

Для компьютеров с IEEE-арифметикой, в которых определены объекты eps, realmax realmin, функция log10 вычисляет следующие величины:

log10(eps)	log10(realmax)	log10(realmin)
-15.6536	308.2547	-307.6527

Сопутствующие функции: EXP, LOG2, LOGM, POW2.

Тригонометрические функции

SIN, SINH **Функции синуса**

Синтаксис:

V= sin(Z)
V = sinh(Z)

Описание:

Функция V = sin(Z) вычисляет синус от значений элементов массива Z.

Функция V = sinh(Z) вычисляет гиперболический синус от значений элементов массива Z.

Массив Z допускает комплексные значения; углы измеряются в радианах.

Для вычисления функции от матрицы следует применять специальные функции funm или expm.

Алгоритм:

Для вычисления функций синуса используются следующие соотношения:

$\sin(x + iy) = \sin(x)\mathrm{ch}(y) + i\cos(x)\mathrm{sh}(y)$;

$\mathrm{sh}(z) = \dfrac{e^z - e^{-z}}{2}$;

$\sin(z) = -i\,\mathrm{sh}(iz)$.

Сопутствующие функции: ASIN, ASINH, CSC, CSCH, ACSC, ACSCH, EXPM, FUNM.

ASIN, ASINH **Функции обратного синуса**

Синтаксис:

V = asin(Z)
V = asinh(Z)

Описание:

Функция V = asin(Z) вычисляет обратную функцию синуса от значений элементов массива Z.

Функция V = sinh(Z) вычисляет обратную функцию гиперболического синуса от значений элементов массива Z.

Массив Z допускает комплексные значения; углы V измеряются в радианах.

Функция Y = asin(X) для действительных значений $-1 \le x \le 1$ определена в интервале $-\pi/2 \le x \le \pi/2$.

Для вычисления функции от матрицы следует применять специальную функцию funm.

Алгоритм:

Для вычисления функций обратного синуса используются следующие соотношения:

$\mathrm{arsh}(z) = \ln[z + (1 + z^2)^{1/2}]$;

$\arcsin(z) = -i\,\mathrm{arsh}(iz)$.

Сопутствующие функции: SIN, SINH, CSC, CSCH, ACSC, ACSCH, FUNM.

CSC, CSCH **Функции косеканса**

Синтаксис:

V = csc(Z)
V = csch(Z)

Описание:

Функция V = csc(Z) вычисляет косеканс от значений элементов массива Z.

Функция V = sinh(Z) вычисляет гиперболический косеканс от значений элементов массива Z.

Массив Z допускает комплексные значения; углы измеряются в радианах.

Для вычисления функции от матрицы следует применять специальную функцию funm.

Алгоритм:

Для вычисления функций косеканса используются следующие соотношения:

csc(z) = 1/sin(z);

csch(z) = 1/sinh(z).

Сопутствующие функции: SIN, SINH, ASIN, ASINH, FUNM.

ACSC, ACSCH **Функции обратного косеканса**

Синтаксис:

V = acsc(Z)

V = acsch(Z)

Описание:

Функция V = acsc(Z) вычисляет обратную функцию косеканса от значений элементов массива Z.

Функция V = acsch(Z) вычисляет обратную функцию гиперболического косеканса от значений элементов массива Z.

Массив Z допускает комплексные значения; углы V измеряются в радианах.

Для вычисления функции от матрицы следует применять специальную функцию funm.

Алгоритм:

Для вычисления функций обратного косеканса используются следующие соотношения:

arccosec(z) = arcsin(1/z);

arcosech(z) = arsinh(1/z).

Сопутствующие функции: SIN, SINH, CSC, CSCH, ASIN, ASINH, FUNM.

COS, COSH **Функции косинуса**

Синтаксис:

V = cos(Z)

V = cosh(Z)

Описание:

Функция V = cos(Z) вычисляет косинус от значений элементов массива Z.

Функция V = cosh(Z) вычисляет гиперболический косинус от значений элементов массива Z.

Массив Z допускает комплексные значения; углы измеряются в радианах.

Для вычисления функции от матрицы следует применять специальные функции funm или expm.

Алгоритм:

Для вычисления функций косинуса используются следующие соотношения:

$\cos(x + iy) = \cos(x)\mathrm{ch}(y) - i\sin(x)\mathrm{sh}(y)$;

$\mathrm{ch}(z) = \dfrac{e^z + e^{-z}}{2}$;

$\cos(z) = \mathrm{ch}(iz)$.

Сопутствующие функции: ACOS, ACOSH, SEC, SECH, ASEC, ASECH, EXPM, FUNM.

ACOS, ACOSH **Функции обратного косинуса**

Синтаксис:

V = acos(Z)
V = acosh(Z)

Описание:

Функция V = acos(Z) вычисляет обратную функцию косинуса от значений элементов массива Z.

Функция V = acosh(Z) вычисляет обратную функцию гиперболического косинуса от значений элементов массива Z.

Массив Z допускает комплексные значения; углы V измеряются в радианах.

Функция Y = acos(X) для действительных значений $-1 \le x \le 1$ определена в интервале $0 \le x \le \pi$.

Для вычисления функции от матрицы следует применять специальную функцию funm.

Алгоритм:

Для вычисления функций обратного косинуса используются следующие соотношения:

$\mathrm{arch}(z) = \ln[z + (z^2 - 1)^{1/2}]$;

$\arccos(z) = -i\,\mathrm{arch}(z)$.

Сопутствующие функции: COS, COSH, SEC, SECH, ASEC, ASECH, FUNM.

SEC, SECH **Функции секанса**

Синтаксис:

V = sec(Z)
V = sech(Z)

Описание:

Функция V = sec(Z) вычисляет секанс от значений элементов массива Z.

Функция V = sech(Z) вычисляет гиперболический секанс от значений элементов массива Z.

Массив Z допускает комплексные значения; углы измеряются в радианах.

Для вычисления функции от матрицы следует применять специальную функцию funm.

Алгоритм:

Для вычисления функций секанса используются следующие соотношения:

sec(z) = 1/cos(z);

sech(z) = 1/cosh(z).

Сопутствующие функции: COS, COSH, ACOS, ACOSH, FUNM.

ASEC, ASECH **Функции обратного секанса**

Синтаксис:

V = asec(Z)

V = asech(Z)

Описание:

Функция V = asec(Z) вычисляет обратную функцию секанса от значений элементов массива Z.

Функция V = asech(Z) вычисляет обратную функцию гиперболического секанса от значений элементов массива Z.

Массив Z допускает комплексные значения; углы V измеряются в радианах.

Для вычисления функции от матрицы следует применять специальную функцию funm.

Алгоритм:

Для вычисления функций обратного секанса используются следующие соотношения:

arcsec(z) = arccos(1/z);

arsech(z) = arcosh(1/z).

Сопутствующие функции: SIN, SINH, CSC, CSCH, ASIN, ASINH, FUNM.

TAN, TANH **Функции тангенса**

Синтаксис:

V = tan(Z)

V = tanh(Z)

Описание:

Функция V = tan(Z) вычисляет тангенс от значений элементов массива Z.

Функция V = sinh(Z) вычисляет гиперболический тангенс от значений элементов массива Z.

Массив Z допускает комплексные значения; углы измеряются в радианах.

Для вычисления функции от матрицы следует применять специальную функцию funm.

Алгоритм:

Для вычисления функций тангенса используются следующие соотношения:

$$\mathrm{tg}(z) = \frac{\sin(z)}{\cos(z)};$$

$$\mathrm{th}(z) = \frac{\sinh(z)}{\cosh(z)}.$$

Сопутствующие функции: ATAN, ATAN2, ATANH, COT, ACOT, COTH, ACOTH, FUNM.

ATAN, ATAN2, ATANH **Функции обратного тангенса**

Синтаксис:

V = atan(Z)
V = atan2(Y, X)
V = atanh(Z)

Описание:

Функция V = atan(Z) вычисляет обратную функцию тангенса от значений элементов массива Z.

Функция V = atan2(Y, X) вычисляет обратную функцию тангенса от значений элементов двух связанных действительных массивов Y и X. Если массивы оказались комплексными, берутся их действительные части. Углы V вычисляются с учетом знаков Y и X и определены в интервале $-\pi \le v \le \pi$.

Функция V = atanh(Z) вычисляет обратную функцию гиперболического тангенса от значений элементов массива Z.

Массив Z допускает комплексные значения; углы V измеряются в радианах.

Для вычисления функции от матрицы следует применять специальную функцию funm.

Алгоритм:

Для вычисления функций обратного тангенса используются следующие соотношения:

$$\mathrm{arth}(z) = \frac{1}{2}\ln\frac{1+z}{1-z};$$

$$\mathrm{arctg}(z) = -i\,\mathrm{arth}(iz);$$

$$\mathrm{Arctg}(y, x) = \begin{cases} arctg\dfrac{y}{x}, & x > 0, \quad -\infty < y < \infty; \\ \pi - arctg\dfrac{y}{|x|}, & x < 0, \quad 0 \le y < \infty; \\ -\pi + arctg\dfrac{|y|}{|x|}, & x < 0, \quad -\infty < y \le 0. \end{cases}$$

Сопутствующие функции: TAN, TANH, COT, COTH, ACOT, ACOTH, FUNM.

COT, COTH **Функции котангенса**

Синтаксис:

V = cot(Z)
V = coth(Z)

Описание:

Функция V = csc(Z) вычисляет котангенс от значений элементов массива Z.

Функция V = sinh(Z) вычисляет гиперболический котангенс от значений элементов массива Z.

Массив Z допускает комплексные значения; углы измеряются в радианах.

Для вычисления функции от матрицы следует применять специальную функцию funm.

Алгоритм:

Для вычисления функций косеканса используются следующие соотношения:

ctg(z) = 1/tg(z);
cth(z) = 1/th(z).

Сопутствующие функции: TAN, TANH, ATAN, ATAN2, ATANH, FUNM.

ACOT, ACOTH **Функции обратного котангенса**

Синтаксис:

V = acot(Z)
V = acoth(Z)

Описание:

Функция V = acsc(Z) вычисляет обратную функцию котангенса от значений элементов массива Z.

Функция V = acsch(Z) вычисляет обратную функцию гиперболического котангенса от значений элементов массива Z.

Массив Z допускает комплексные значения; углы V измеряются в радианах.

Для вычисления функции от матрицы следует применять специальную функцию funm.

Алгоритм:

Для вычисления функций обратного котангенса используются следующие соотношения:

$$\mathrm{arcth}(z) = \frac{1}{2}\ln\frac{z+1}{z-1};$$

$$\mathrm{arcctg}(z) = i\,\mathrm{arcth}(iz).$$

Сопутствующие функции: TAN, TANH, COT, COTH, ATAN, ATAN2, ATANH, FUNM.

Преобразования систем координат

CART2POL

Преобразование декартовой системы координат в полярную и цилиндрическую

Синтаксис:

[TH, R] = cart2pol(X, Y)
[TH, R, Z] = cart2pol(X, Y, Z)

Описание:

Функция [TH, R] = cart2pol(X, Y) преобразовывает точки декартовой системы координат в точки полярной системы координат. Размеры массивов X и Y должны быть согласованы. Угол TH измеряется в радианах.

Функция [TH, R, Z] = cart2pol(X, Y, Z) преобразовывает точки трехмерной декартовой системы координат в точки цилиндрической системы координат. Размеры массивов X, Y и Z должны быть согласованы. Угол TH измеряется в радианах.

Алгоритм:

Для вычисления используются следующие формулы преобразования:
r = sqrt(x.^2 + y.^2);
th = atan2(y, x).

Сопутствующие функции: POL2CART.

CART2SPH

Преобразование декартовой системы координат в сферическую

Синтаксис:

[AZ, EL, R] = cart2sph(X, Y, Z)

Описание:

Функция [AZ, EL, R] = cart2sph(X, Y, Z) преобразовывает точки трехмерной декартовой системы координат в точки сферической системы координат. Размеры массивов X, Y и Z должны быть согласованы. Углы AZ, EL измеряются в радианах.

Алгоритм:

Для вычисления используются следующие формулы преобразования:
r = sqrt(x.^2 + y.^2 + z.^2);
elev = atan2(z, sqrt(x.^2 + y.^2));
az = atan2(y, x).

Сопутствующие функции: SPH2CART.

POL2CART

Преобразование полярной и цилиндрической систем координат в декартову

Синтаксис:

[X, Y] = pol2cart(TH, R)
[X, Y, Z] = pol2cart(TH, R, Z)

Описание:

Функция [X, Y] = pol2cart(TH, R) преобразовывает точки полярной системы координат в точки декартовой системы координат. Размеры массивов X и Y должны быть согласованы. Угол TH измеряется в радианах.

Функция [TH, R, Z] = cart2pol(X, Y, Z) преобразовывает точки цилиндрической системы координат в точки трехмерной декартовой системы координат. Размеры массивов X, Y и Z должны быть согласованы. Угол TH измеряется в радианах.

Алгоритм:

Для вычисления используются следующие формулы преобразования:

x = r.*cos(th);

y = r.*sin(th);

z = z.

Сопутствующие функции: CART2POL.

SPH2CART **Преобразование сферической системы координат в декартову**

Синтаксис:

[X, Y, Z] = sph2cart(AZ, EL, R)

Описание:

Функция [X, Y, Z] = sph2cart(AZ, EL, R) преобразовывает точки сферической системы координат в точки декартовой системы координат. Размеры массивов X, Y и Z должны быть одинаковы. Углы AZ, EL измеряются в радианах.

Алгоритм:

Для вычисления используются следующие формулы преобразования:

z = r .* sin(elev);

x = r .* cos(elev) .* cos(az);

y = r .* cos(elev) .* sin(az).

Сопутствующие функции: CART2SPH, CART2POL, POL2CART.

Специальные функции

Решение многих научных и технических проблем связано с исследованием специальных функций. Входящие в состав системы MATLAB 5 специальные функции соответствуют справочнику по специальным функциям [1]. Они включают функции Бесселя и Эйри, бета-функции, эллиптические функции Якоби и полные эллиптические интегралы, функции вероятностей, гамма-функции, интегральную показательную функцию и присоединенную функцию Лежандра.

BESSELJ, BESSELY **Функции Бесселя**

Синтаксис:

J = besselj(nu, Z)
J = besselj(nu, Z, 1)
[J, ierr] = besselj(nu, Z)

Y = bessely(nu, Z)
Y = bessely(nu, Z, 1)
[Y, ierr] = bessely(nu, Z)

Описание:

Линейное дифференциальное уравнение вида

$$z^2\frac{d^2y}{dz^2}+z\frac{dy}{dz}+(z^2-\nu^2)y=0,$$

где ν - неотрицательная величина, называется *уравнением Бесселя*, а его решения известны как *функции Бесселя первого рода* $J_\nu(z)$ и *второго рода* $Y_\nu(z)$.

Функции Бесселя *первого рода* $J_\nu(z)$ и *второго рода* $Y_\nu(z)$ определяются следующим образом:

$$J_\nu(z)=\left(\frac{z}{2}\right)^2\sum_{k=0}^{\infty}\frac{(-z^2/4)^k}{k!\,\Gamma(\nu+k+1)},\qquad \text{где } \Gamma(\alpha)=\int_0^{\infty}e^{-t}t^{\alpha-1}dt;$$

$$Y_\nu(z)=\frac{J_\nu(z)\cos(\nu\pi)-J_{-\nu}(z)}{\sin(\nu\pi)}.$$

Эти функции линейно независимы и задают полное множество решений для уравнения Бесселя.

Функция J = besselj(nu, Z) вычисляет функции Бесселя первого рода для каждого элемента комплекснозначного массива Z.

Функция Y = bessely(nu, Z) вычисляет функции Бесселя второго рода для действительных неотрицательных значений nu и аргумента Z.

Порядок функции nu может принимать любые действительные значения, не обязательно целочисленные. Результат является действительным для положительных элементов массива Z.

Если входные массивы nu и Z имеют одинаковые размеры, то результатом является массив того же размера. Если один из массивов скаляр, то он расширяется до размеров второго массива. Если один из массивов вектор-строка, а второй вектор-столбец, то результатом является двумерная таблица значений соответствующей функции Бесселя.

Функции J = besselj(nu, Z, 1) и Y = bessely(nu, Z) вычисляют функции Бесселя первого рода, масштабированные множителем exp(-abs(imag(z))), для каждого элемента комплекснозначного массива Z.

Функции [J, ierr] = besselj(nu, Z) и [Y, ierr] = bessely(nu, Z) возвращают также сообщение об ошибке, как это описано в следующей таблице.

ierr	*Описание*
1	Неправильные аргументы
2	Переполнение, результат Inf
3	Некоторая потеря точности при уменьшении параметра
4	Полная потеря точности, z или nu слишком велики
5	Нет сходимости, результат NaN

Примеры:

Построить линии уровней для модуля и фазы функции Бесселя первого рода порядка 0.

```
[X, Y] = meshgrid(-4 : 0.025 : 4,  -1.5 : 0.025 : 1.5);
    H = besselj(0, X + i*Y);
contour(X, Y, abs(H), 0 : 0.2 : 3.2), hold on
contour(X, Y, (180/pi)*angle(H), -180 : 10 : 180); hold off
```

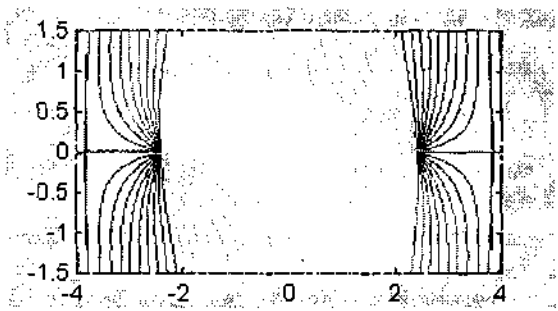


Рис. 6.1

Построить линии уровней для модуля и фазы функции Бесселя второго рода порядка 0.

```
[X, Y] = meshgrid(-4 : 0.025 : 4,  -1.5 : 0.025 : 1.5);
    H = bessely(0, X + i*Y);
    contour(X, Y, abs(H), 0 : 0.2 : 3.2), hold on
    contour(X, Y, (180/pi)*angle(H), -180 : 10 : 180); hold off
```

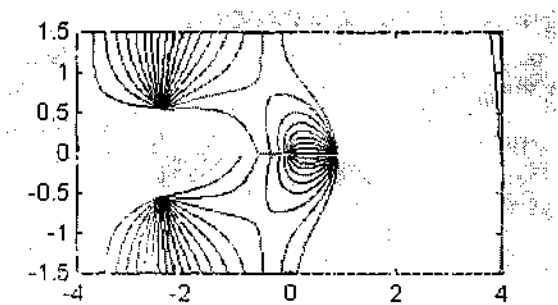


Рис. 6.2

Алгоритм:

М-функции besselj и bessely используют интерфейс смешанного программирования (mex-интерфейс) для вызова библиотеки на языке Fortran, описанной в работах [2,3].

Сопутствующие функции: AIRY, BESSELH, BESSELI, BESSELK.

Ссылки:

1. Справочник по специальным функциям с формулами, графиками и таблицами/ Под ред. М. Абрамовича и И. Стиган: Пер. с англ. М.: Наука, 1979. 832 с.
2. Carrier, Krook, Pearson. *Functions of a Complex Variable: Theory and Technique.* N. Y.: Hod Books, 1983.
3. Amos D. E. *A Subroutine Package for Bessel Functions of a Complex Argument and Nonnegative Order.* Sandia National Laboratory Report, SAND85-1018, May, 1985.

BESSELI, BESSELK **Модифицированные функции Бесселя**

Синтаксис:

I = besseli(nu, Z)
I = besseli(nu, Z, 1)
[I, ierr] = besseli(nu, Z)

K = besselk(nu, Z)
K = besselk(nu, Z, 1)
[K, ierr] = besselk(nu, Z)

Описание:

Линейное дифференциальное уравнение вида

$$z^2 \frac{d^2 y}{dz^2} + z\frac{dy}{dz} - (z^2 + \nu^2)\, y = 0\,,$$

где ν - неотрицательная величина, называется *модифицированным уравнением Бесселя*, а его решения известны как *модифицированные функции Бесселя первого* $I_\nu(z)$ и *второго рода* $K_\nu(z)$.

Модифицированные функции Бесселя *первого рода* $I_\nu(z)$ и *второго рода* $K_\nu(z)$ определяются следующим образом:

$$I_\nu(z) = \left(\tfrac{z}{2}\right)^2 \sum_{k=0}^{\infty} \frac{(z^2/4)^k}{k!\,\Gamma(\nu+k+1)}\,, \qquad \text{где } \Gamma(a) = \int_0^\infty e^{-t} t^{a-1} dt\,;$$

$$K_\nu(z) = \pi/2 \frac{I_{-\nu}(z) - I_\nu(z)}{\sin(\nu\pi)}\,.$$

Эти функции линейно независимы и задают полное множество решений для модифицированного уравнения Бесселя.

Функция I = besseli(nu, Z) вычисляет модифицированные функции Бесселя первого рода для каждого элемента комплекснозначного массива Z.

Функция K = besselk(nu, Z) вычисляет модифицированные функции Бесселя второго рода для действительных неотрицательных значений nu и аргумента Z.

Порядок функций nu может принимать любые действительные значения, не обязательно целочисленные. Результат является действительным для положительных элементов массива Z.

Если входные массивы nu и Z имеют одинаковые размеры, то результатом является массив того же размера. Если один из массивов скаляр, то он расширяется до размеров второго массива. Если один из массивов вектор-строка, а второй вектор-столбец, то результатом является двумерная таблица значений соответствующей модифицированной функции Бесселя.

Функции I = besseli(nu, Z, 1) и K = besselk(nu, Z, 1) вычисляют модифицированные функции Бесселя, масштабированные множителем exp(-abs(real(z))), для каждого элемента комплекснозначного массива Z.

Функции [I, ierr] = besseli(nu, Z) и [K, ierr] = besselk(nu, Z) возвращают также сообщение об ошибке, как это описано в следующей таблице.

ierr	*Описание*
1	Неправильные аргументы
2	Переполнение, результат Inf
3	Некоторая потеря точности при уменьшении параметра
4	Полная потеря точности, z или nu слишком велики
5	Нет сходимости, результат NaN

Примеры:

Построить линии уровней для модуля и фазы модифицированной функции Бесселя первого рода порядка 0.

```
[X, Y] = meshgrid(-4 : 0.025 : 4,  -1.5 : 0.025 : 1.5);
    H = besseli(0, X + i*Y);
contour(X, Y, abs(H), 0 : 0.2 : 3.2), hold on
contour(X, Y, (180/pi)*angle(H), -180 : 10 : 180); hold off
```

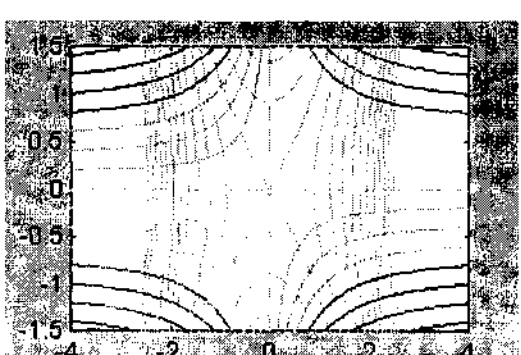


Рис. 6.3

Построить линии уровней для модуля и фазы модифицированной функции Бесселя второго рода порядка 0.

```
[X, Y] = meshgrid(-4 : 0.025 : 4,  -1.5 : 0.025 : 1.5);
    H = besselk(0, X + i*Y);
    contour(X, Y, abs(H), 0 : 0.2 : 3.2),  hold on
    contour(X, Y, (180/pi)*angle(H),  -180 : 10 : 180);  hold off
```

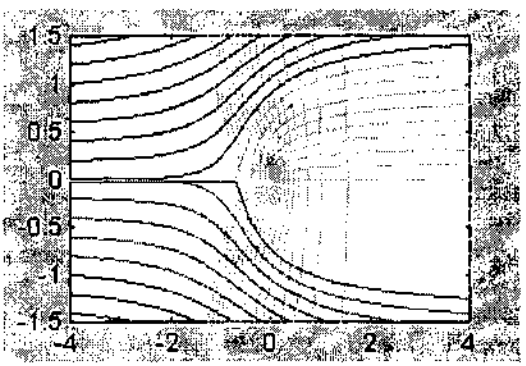


Рис. 6.4

Алгоритм:

М-функции besseli и besselk используют интерфейс смешанного программирования (mex-интерфейс) для вызова библиотеки на языке Fortran, описанной в работах [1, 2].

Сопутствующие функции: AIRY, BESSELH, BESSELJ, BESSELY.

Ссылки:

1. Carrier, Krook, Pearson. *Functions of a Complex Variable: Theory and Technique.* Hod Books, 1983.
2. Amos D. E. *A Subroutine Package for Bessel Functions of a Complex Argument and Nonnegative Order.* Sandia National Laboratory Report, SAND85-1018, May, 1985.

BESSELH **Функции Ханкеля**

Синтаксис:

```
H = besselh(nu, k, Z)          H = besselh(nu, k, Z, 1)
H = besselh(nu, Z)             [H, ierr] = besselh(...)
```

Описание:

Функции Ханкеля, или функции Бесселя *третьего рода*, $H_\nu^{(1)}(z)$ и $H_\nu^{(2)}(z)$ являются линейными комбинациями функций Бесселя первого и второго рода и определяются следующим образом:

$$H_\nu^{(1)}(z) = J_\nu(z) + iY_\nu(z);$$

$$H_\nu^{(2)}(z) = J_\nu(z) - iY_\nu(z).$$

Эти функции линейно независимы и задают полное множество решений для уравнения Бесселя.

Функция H = besselh(nu, k, Z) вычисляет для значений k = 1, 2 функции Ханкеля порядка nu для каждого элемента комплекснозначного массива Z.

Функция H = besselh(nu, Z) вычисляет функцию Ханкеля порядка nu, принимая по умолчанию значение k = 1.

Функция H = besselh(nu, 1, Z, 1) вычисляет функцию Ханкеля $H_\nu^{(1)}(z)$ порядка nu, масштабированную множителем exp(-iz).

Функция H = besselh(nu, 2, Z, 1) вычисляет функцию Ханкеля $H_\nu^{(2)}(z)$ порядка nu, масштабированную множителем exp(iz).

Порядок функций nu может принимать любые действительные значения, не обязательно целочисленные. Результат является действительным для положительных элементов массива Z.

Если входные массивы nu и Z имеют одинаковые размеры, то результатом является массив того же размера. Если один из массивов скаляр, то он расширяется до размеров второго массива. Если один из массивов вектор-строка, а второй вектор-столбец, то результатом является двумерная таблица значений соответствующей функции Ханкеля.

Функции в форме [H, ierr] = besselh(...) возвращают также сообщение об ошибке, как это описано в следующей таблице.

ierr	*Описание*
1	Неправильные аргументы
2	Переполнение, результат Inf
3	Некоторая потеря точности при уменьшении параметра
4	Полная потеря точности, z или nu слишком велики
5	Нет сходимости, результат NaN

Примеры:

Построить линии уровней для модуля и фазы функции Ханкеля $H_0^{(1)}(z)$.

```
[X, Y] = meshgrid(-4 : 0.025 : 4,  -1.5 : 0.025 : 1.5);
    H = besselh(0, 1, X + i*Y);
contour(X, Y, abs(H), 0 : 0.2 : 3.2), hold on
contour(X, Y, (180/pi)*angle(H), -180 : 10 : 180); hold off
```

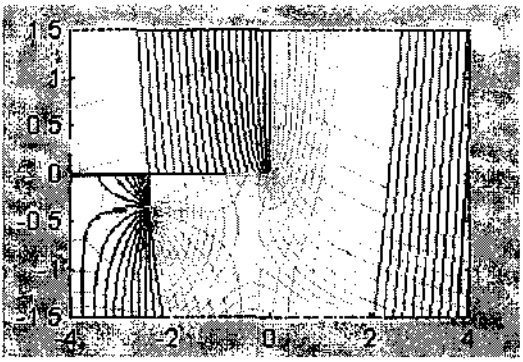


Рис. 6.5

Построить линии уровней для модуля и фазы функции Ханкеля $H_0^{(2)}(z)$

```
[X, Y] = meshgrid(-4 : 0.025 : 4,  -1.5 : 0.025 : 1.5);
    H = besselh(0, 2, X + i*Y);
    contour(X, Y, abs(H), 0 : 0.2 : 3.2),  hold on
    contour(X, Y, (180/pi)*angle(H),  -180 : 10 : 180);  hold off
```

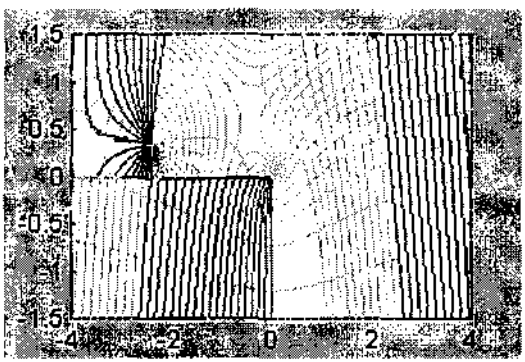


Рис. 6.6

Алгоритм:

М-функция besselh использует интерфейс смешанного программирования (mex-интерфейс) для вызова библиотеки на языке Fortran, описанной в работах [1, 2].

Сопутствующие функции: AIRY, BESSELI, BESSELJ, BESSELK, BESSELY.

Ссылки:

1. Carrier, Krook, Pearson. *Functions of a Complex Variable: Theory and Technique.* Hod Books, 1983.
2. Amos D. E. *A Subroutine Package for Bessel Functions of a Complex Argument and Nonnegative Order.* Sandia National Laboratory Report, SAND85-1018, May, 1985.

AIRY **Функции Эйри**

Синтаксис:

W = airy(Z)
W = airy(k, Z)
[W, ierr] = airy(k, Z)

Описание:

Функции Эйри формируют пару независимых решений для дифференциального уравнения

$$\frac{d^2w}{dz^2} - zw = 0 .$$

Эти решения известны как *функции Эйри первого* Ai(z) и *второго* Bi(z) *рода.*

Связь функций Эйри с модифицированными функциями Бесселя первого рода задается соотношениями

$$\mathrm{Ai}(z) = \frac{\sqrt{z}}{3}(\mathrm{I}_{-1/3}(\zeta) - \mathrm{I}_{1/3}(\zeta)) ;$$

$$\mathrm{Bi}(z) = \sqrt{\frac{z}{3}}(\mathrm{I}_{-1/3}(\zeta) + \mathrm{I}_{1/3}(\zeta)) , \text{ где } \zeta = \frac{2}{3}z^{3/2} .$$

Функция W = airy(Z) вычисляет функцию Эйри Ai(Z) для каждого элемента комплекснозначного массива Z.

Функция W = airy(k, Z) возвращает разные результаты в зависимости от значения параметра k в соответствии со следующей таблицей.

k	*Описание*
0	Вычисление функции первого рода Эйри Ai(z)
1	Вычисление производной функции Ai(z)
2	Вычисление функции второго рода Эйри Bi(z)
3	Вычисление производной функции Bi(z)

Функция [W,ierr] = airy(k,Z) возвращает значение флага ошибки в соответствии со следующей таблицей.

ierr	*Описание*
1	Неправильные аргументы
2	Переполнение, результат Inf
3	Некоторая потеря точности при уменьшении параметра
4	Полная потеря точности, z или nu слишком велики
5	Нет сходимости, результат NaN

Примеры:

Построить линии уровней для модуля и фазы функции Эйри первого рода Ai(z).

```
[X, Y] = meshgrid(-4 : 0.025 : 4,  -1.5 : 0.025 : 1.5);
    H = airy(0, X + i*Y);
     contour(X, Y, abs(H), 0 : 0.2 : 3.2), hold on
              contour(X, Y, (180/pi)*angle(H), -180 : 10 : 180); hold off
```

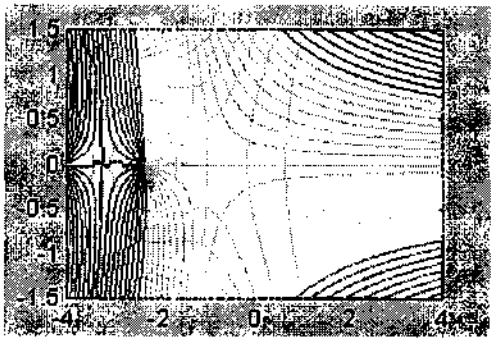


Рис. 6.7

Построить линии уровней для модуля и фазы функции Эйри второго рода Bi(z).

```
[X, Y] = meshgrid(-4 : 0.025 : 4,   -1.5 : 0.025 : 1.5);
    H = airy(2, X + i*Y);
    contour(X, Y, abs(H), 0 : 0.2 : 3.2),  hold on
    contour(X, Y, (180/pi)*angle(H),  -180 : 10 : 180);  hold off
```

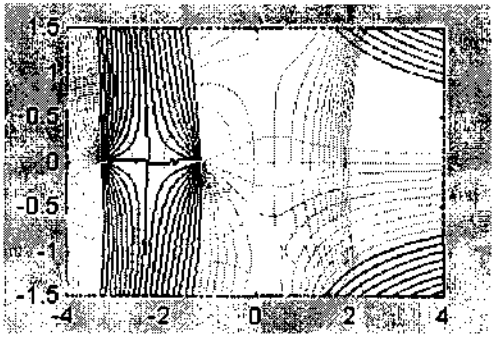


Рис. 6.8

Алгоритм:

М-функция airy использует интерфейс смешанного программирования (mex-интерфейс) для вызова библиотеки на языке Fortran, описанной в работах [1, 2].

Сопутствующие функции: BESSELI, BESSELJ, BESSELK, BESSELY.

Ссылки:
1. Amos D. E. A Subroutine Package for Bessel Functions of a Complex Argument and Nonnegative Order. Sandia National Laboratory Report, SAND85-1018, May, 1985.
2. Amos D. E. A Portable Package for Bessel Functions of a Complex Argument and Nonnegative Order. Trans. Math. Software, 1986.

BETA, BETAINC, BETALN

Бета-функции

Синтаксис:

```
B = beta(Z, W)
I = betainc(X, Z, W)
y = betaln(Z, W)
```

Описание:

Полная бета-функция B = beta(z, w) для комплексных аргументов z и w определяется следующим образом [1]:

$$B(z, w) = \int_0^1 t^{z-1}(1-t)^{w-1}\,dt = \frac{\Gamma(z)\Gamma(w)}{\Gamma(z+w)}.$$

Неполная бета-функция B = betainc(p, q) определяется так [1]:

$$I_x(z, w) = \frac{1}{B(z,w)}\int_0^x t^{z-1}(1-t)^{w-1}\,dt.$$

Функция y = betaln(z, w) представляет натуральный логарифм полной бета-функции B = beta(z, w). Поскольку бета-функция изменяется в широких пределах, знание ее логарифма может оказаться более предпочтительным. Вычисление функции ln(B) реализуется без вычисления функции beta(p, q).

Если Z и W - массивы, то их размеры должны быть согласованы, а вычисления выполняются для соответствующих пар элементов.

Алгоритм:

```
betaln(z, w) = gammaln(z) + gammaln(w) - gammaln(z + w)
beta(z, w) = exp(betaln(z, w))
```

Пример:

```
format rat
beta((0:10)', 3)
ans =
    0/0
    1/3
    1/12
    1/30
    1/60
    1/105
    1/168
    1/252
    1/360
    1/495
    1/660
```

В данном случае при целочисленных аргументах

beta(n, 3) = (n - 1)! * 2!/(n + 2)! = 2/((n * (n + 1) * (n + 2))

и представляет собой отношение двух сравнительно небольших целых чисел, так что результат в формате format rat оказывается точным.

Для x = 510 betaln(x, x) = -708.8616, что для компьютеров с IEEE-арифметикой немного меньше, чем log(realmin) = -708.3964, так что при вычислении функции beta(x, x) следует проявлять осторожность.

Сопутствующие функции: GAMMA, GAMMAINC, GAMMALN.

Ссылки:

1. Справочник по специальным функциям с формулами, графиками и таблицами / Под ред. М. Абрамовича и И. Стиган: Пер. с англ. М.: Наука, 1979. 832 с.

ELLIPJ **Эллиптические функции Якоби**

Синтаксис:

[SN, CN, DN] = ellipj(M, Z)
[SN, CN, DN] = ellipj(M, Z, tol)

Определения:

Введем интеграл

$$z = \int_0^{\varphi} \frac{d\theta}{(1 - m\sin^2\theta)^{1/2}};$$

функция φ = am(z) называется *амплитудой*, функция sn(z) - *синусом амплитуды*, функция cn(z) - *косинусом амплитуды*, и dn(z) - *дельтой амплитуды*.

Эти эллиптические функции Якоби связаны между собой следующим образом [1]:

$$sn(z) = \sin(\varphi), \quad cn(z) = \cos(\varphi), \quad dn(z) = (1 - m\, sn(z))^{1/2}.$$

Описание:

Функция [SN, CN, DN] = ellipj(M, Z) вычисляет эллиптические функции Якоби sn, cn, dn для заданных значений M и Z, которые могут быть как скалярами, так и массивами, но обязательно одинаковых размеров.

Функция [SN, CN, DN] = ellipj(M, Z, tol) вычисляет эллиптические функции Якоби с заданной точностью tol. По умолчанию это значение равно константе eps = 2.220446049250313e-016. Увеличение этого значения сокращает время вычислений, но приводит к потере точности.

Алгоритм:

Функция ellipj вычисляет эллиптические функции Якоби, используя метод арифметико-геометрического среднего [1] и следующие начальные значения параметров:

$a_0 = 1; \quad b_0 = (1 - m)^{1/2}; \quad c_0 = m^{1/2}.$

Реализуется следующий итерационный процесс:

$$a_i = \frac{1}{2}(a_{i-1} + b_{i-1});$$

$$b_i = (a_{i-1} b_{i-1})^{1/2};$$

$$c_i = \frac{1}{2}(a_{i-1} - b_{i-1}),$$

который заканчивается на шаге N, когда величиной c_N можно пренебречь в пределах заданной точности tol.

Затем для заданного z вычисляется амплитуда φ_N в радианах по формуле $\varphi_N = 2^N a_N z$ и последовательно значения φ_{N-1}, φ_{N-2}, ..., φ_1, φ_0, используя рекуррентное соотношение

$$\sin(2\varphi_{i-1} - \varphi_i) = \frac{c_i}{a_i}\sin(\varphi_i).$$

Тогда

$$\mathrm{sn}(z, m) = \sin(\varphi_0), \quad \mathrm{cn}(z, m) = \cos(\varphi_0), \quad \mathrm{dn}(z, m) = \frac{\cos(\varphi_0)}{\cos(\varphi_1 - \varphi_0)}.$$

Ограничения:

Входной параметр m должен принадлежать диапазону $0 \le m \le 1$. Способы приведения параметра m к этому диапазону описаны в работе [1].

Пример:

Построить эллиптические функции Якоби sn(z, 1/2), cn(z, 1/2) и dn(z, 1/2) в диапазоне от 0 до 4K, где четвертьпериод K равен ellipke(1/2).

```
Z = 0 : 0.05 : ellipke(1/2)*4;
M = 1/2*ones(size(Z));
[Sn, Cn, Dn] = ellipj(Z, M);
plot(Z, Sn), grid, hold on, plot(Z, Cn, '--r'), plot(Z, Dn, ':b')
gtext('sn'), gtext('cn'), gtext('dn')
```

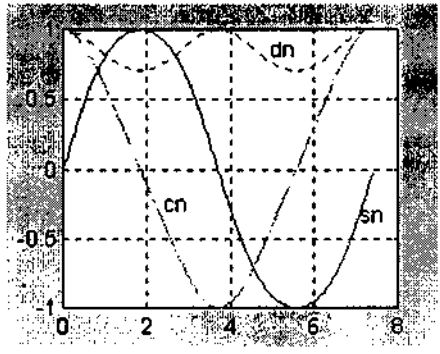


Рис. 6.9

Сопутствующие функции: ELLIPKE.

Ссылки:

1. Справочник по специальным функциям с формулами, графиками и таблицами / Под ред. М. Абрамовича и И. Стиган: Пер. с англ. М.: Наука, 1979. 832 с.

ELLIPKE **Полные эллиптические интегралы**

Синтаксис:

K = ellipke(M)
[K, E] = ellipke(M)
[K, E] = ellipke(M, tol)

Определения:

Полные эллиптические интегралы первого и второго рода определяются следующим образом [1]:

$$K(m) = \int_0^{\pi/2} \frac{d\theta}{(1 - m\sin^2\theta)^{1/2}};$$

$$E(m) = \int_0^{\pi/2} (1 - m\sin^2\theta)^{1/2} d\theta.$$

Описание:

Функция K = ellipke(M) вычисляет полный эллиптический интеграл первого рода для заданных значений элементов массива M.

Функция [K, E] = ellipke(M) вычисляет полные эллиптические интегралы первого и второго рода для каждого элемента массива M.

Функция [K, E] = ellipke(M, tol) выполняет вычисления с заданной точностью tol. По умолчанию это значение равно константе eps = 2.220446049250313e-016. Увеличение этого значения сокращает время вычислений, но приводит к потере точности.

Алгоритм:

Функция ellipke вычисляет эллиптические интегралы, используя метод арифметико-геометрического среднего [1] и следующие начальные значения параметров:

$a_0 = 1$; $b_0 = (1 - m)^{1/2}$; $c_0 = m^{1/2}$.

Реализуется следующий итерационный процесс:

$$a_i = \frac{1}{2}(a_{i-1} + b_{i-1});$$

$$b_i = (a_{i-1}b_{i-1})^{1/2};$$

$$c_i = \frac{1}{2}(a_{i-1} - b_{i-1}),$$

который заканчивается на шаге N, когда величиной c_N можно пренебречь в пределах заданной точности tol.

Полные эллиптические интегралы первого и второго рода вычисляются по следующим формулам:

$$K(m) = \frac{\pi}{2a_N};$$

$$E(m) = (1 - \gamma)K(m),$$

где $\gamma = \frac{1}{2}\sum_{i=0}^{N} 2^i c_i^2$.

Пример:

Вычислить и построить графики полных эллиптических интегралов первого и второго рода для значений m в диапазоне от 0 до 1.

```
M = 0 : 0.05 : 1;
[K, E] = ellipke(M);
plot(M, K), grid, hold on, plot(M, E, '--r')
gtext('K'), gtext('E')
```

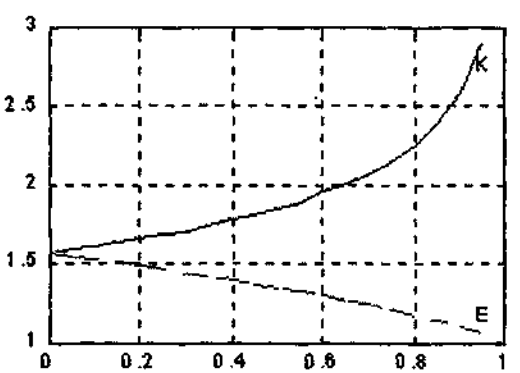


Рис. 6.10

Ограничения:

Входной параметр m должен принадлежать диапазону $0 \le m \le 1$.

Сопутствующие функции: ELLIPJ.

Ссылки:

1. Справочник по специальным функциям с формулами, графиками и таблицами/ Под ред. М. Абрамовича и И. Стиган: Пер. с англ. М.: Наука, 1979. 832 с.

ERF, ERFC, ERFCX, ERFINV **Интегралы вероятностей**

Синтаксис:

Y = erf(X)
Y = erfc(X)
Y = erfcx(X)
X = erfinv(Y)

Определения:

Интеграл вероятностей, или функция ошибок, erf(x) определяется следующим образом [2]:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt .$$

Функция, дополнительная к интегралу вероятностей, y = erfc(x) называется *дополнительным интегралом вероятностей* [2] и задается соотношением

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt = 1 - \mathrm{erf}(x).$$

Нормированный дополнительный интеграл вероятностей y = erfcx(x) определяется по формуле

$$\mathrm{erfcx}(x) = e^{x^2} \mathrm{erfc}(x).$$

Функция, обратная интегралу вероятностей, x = erfinv(y) имеет область определения -1 < y < 1 и область значений -Inf < x < Inf, причем erfinv(-1) = -Inf, erfinv(1) = Inf, а для значений abs(y)>1 erfinv(y) = NaN.

Описание:

Функция Y = erf(X) возвращает значения интеграла вероятностей для всех элементов действительного массива X.

Функция Y = erfc(X) вычисляет значения дополнительного интеграла вероятностей.

Функция Y = erfcx(X) вычисляет значения нормированного дополнительного интеграла вероятностей.

Функция X = erfinv(Y) возвращает значения функции, обратной к интегралу вероятностей, для всех элементов действительного массива Y, удовлетворяющих условию -1 < y < 1.

Алгоритм:

Алгоритм основан на работе [1] и представляет собой переработанную с языка Fortran программу из раздела NETLIB/SPECFUN, написанную W. J. Cody (Argonne National Laboratory, March 19, 1990).

При вычислении функции, обратной к интегралу вероятностей, для получения начального приближения используются рациональные аппроксимации, имеющие точность до шести значащих цифр. Затем это приближение используется для получения точного результата с помощью двух шагов метода Ньютона. Соответствующий М-файл можно модифицировать, чтобы отказаться от уточняющих итераций по методу Ньютона. Возникающий код приблизительно в 3 раза ускоряет выполнение, но значительно теряет в точности.

Пример:

Вычислить и построить график нормальной функции распределения, если известен интеграл вероятностей.

```
x = -3:0.1:3;
stand = (1 + (erf(x/sqrt(2))))./2;
plot(x, stand)
```

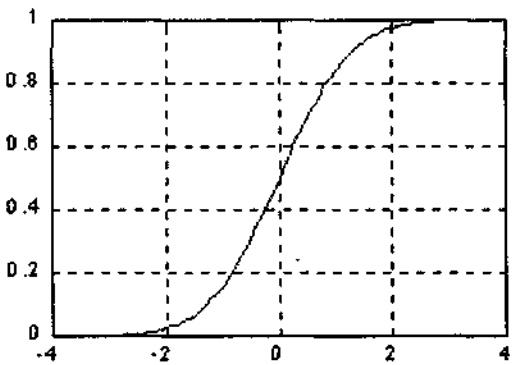


Рис. 6.11

Сопутствующие функции: HTML-справка.

Ссылки:

1. Cody W. J. Rational Chebyshev approximations for the error function//*Math. Comp.*, 1969. P. 631-638.
2. Корн Г., Корн Т. *Справочник по математике для научных работников и инженеров.* М.: Наука, 1968. 720 с.

GAMMA, GAMMAINC, GAMMALN **Гамма-функции**

Синтаксис:

```
Y = gamma(A)
Y = gammainc(X, A)
Y = gammaln(A)
```

Определения:

Гамма-функция gamma(a) определяется следующим образом [3]:

$$\text{gamma}(a) = \Gamma(a) = \int_0^{\infty} e^{-t} t^{a-1} dt.$$

Неполная гамма-функция gammainc(x, a) задается соотношением

$$\text{gammainc}(x, a) = P(x, a) = \frac{1}{\Gamma(a)} \int_0^{x} e^{-t} t^{a-t} dt.$$

Логарифмическая гамма-функция определяется так:

gammaln(a) = ln Γ(a).

Описание:

Функция Y = gamma(A) возвращает значения гамма-функции для каждого значения элемента действительного массива A.

Функция Y = gammainc(X, A) возвращает значения неполной гамма-функции для каждой пары элементов действительных массивов X и A.

Функция Y = gammaln(A) возвращает значения логарифмической гамма-функции для каждого значения элемента действительного массива A. Эта функция используется для того, чтобы избежать переполнения разрядной сетки, которое может возникнуть при прямом вычислении значений log(gamma(A)).

Пример:

```
[X, A] = meshgrid(0 : 0.05 : 5,  0 : 0.05 : 4);
    H = X.^(-A).*gammainc(X, A);
    C = contour(X, AX, H, -6 : 0.5 : 6);
grid, clabel(C, [0.2 0.6 1]
```

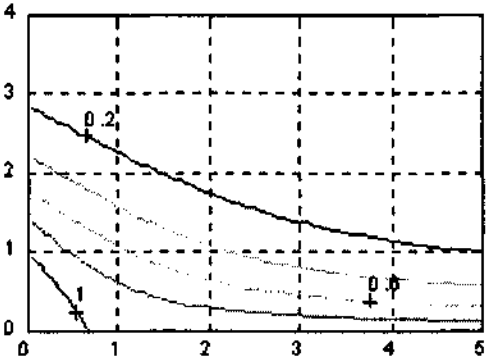


Рис. 6.12

Алгоритм:

Алгоритм основан на работе [1] и представляет собой переработанную с языка Fortran программу из раздела NETLIB/SPECFUN, написанную W. J. Cody (Argonne National Laboratory, October 12, 1989).

Вычисление неполной гамма-функции основано на рекуррентных соотношениях, описанных в работе [2].

Сопутствующие функции: ERF, ERFCORE, ERFC, ERFCX, ERFINV.

Ссылки:

1. Cody W. J. An Overview of Software Development for Special Functions//*Lecture Notes in Mathematics*. Berlin, 1976. Vol. 506.
2. Справочник по специальным функциям с формулами, графиками и таблицами / Под ред. М. Абрамовича и И. Стиган: Пер. с англ. М.: Наука, 1979. 832 с.
3. Корн Г., Корн Т. *Справочник по математике для научных работников и инженеров.* М.: Наука, 1968. 720 с.

EXPINT — Интегральная показательная функция

Синтаксис:

Y = expint(X)

Определения:

Интегральная показательная функция определяется следующим образом [1]:

$$E_1(x) = \int_x^{\infty} \frac{e^{-t}}{t} dt .$$

Другое определение интегральной показательной функции связано с определением главного значения интеграла в смысле Коши [2]

$$Ei(x) = vp \int_{-\infty}^{x} \frac{e^{t}}{t} dt ,$$

который для положительных значений x связан с функцией E1(x) следующим образом:

$E_1(-x + i*0) = -Ei(x) - i*\pi$;

$Ei(x) = Re(-E_1(-x))$.

Описание:

Функция Y = expint(X) возвращает значения интегральной показательной функции для каждого значения элемента действительного массива X.

Алгоритм:

Для значений x в диапазоне [-38, 2] используется разложение в ряд

$$E_1(x) = -\gamma - \ln(x) - \sum_{n=1}^{\infty} \frac{(-1)^n x^n}{n n!} ;$$

для остальных значений x используется представление в виде цепной дроби

$$E_1(x) = e^{-x}\left(\frac{1}{x+}\frac{1}{1+}\frac{1}{x+}\frac{2}{1+}\frac{2}{x+}\ldots\right).$$

Пример:

```
x = 0.02 : 0.01 : 2;
E1 = expint(x);
Ei = real(-expint(-x));
plot(x, E1), grid, hold on, plot(x, Ei)
gtext('E1'), gtext('Ei')
```

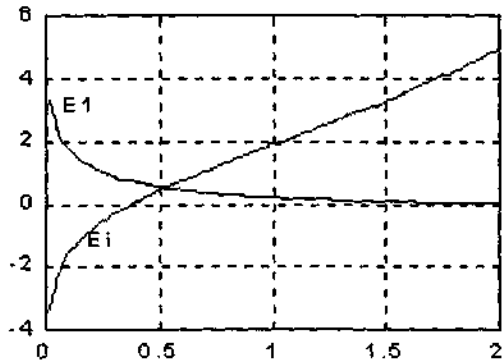


Рис. 6.13

Сопутствующие функции: HTML-справка.

Ссылки:

1. Справочник по специальным функциям с формулами, графиками и таблицами/ Под ред. М. Абрамовича и И. Стиган: Пер. с англ. М.: Наука, 1979. 832 с.
2. Корн Г., Корн Т. *Справочник по математике для научных работников и инженеров.* М.: Наука, 1968. 720 с.

LEGENDRE **Присоединенные функции Лежандра**

Синтаксис:

```
P = legendre(n, X)
S = legendre(n, X, 'sch')
```

Определения:

Присоединенные ненормализованные функции Лежандра определяются через полиномы Лежандра следующим образом [1]:

$$P_n^m(x) = (-1)^m (1-x^2)^{m/2} \frac{d^m}{dx^m} P_n(x),$$

где $P_n(x)$ - полином Лежандра степени n вида

$$P_n(x) = \frac{1}{2^n n!}\left[\frac{d^n}{dx^n}(x^2-1)^n\right].$$

Полунормализованные по Шмидту присоединенные функции Лежандра $S_n^m(x)$ связаны с ненормализованными присоединенными функциями Лежандра $P_n^m(x)$ следующим соотношением:

$$S_n^m(x) = \sqrt{\frac{2(n-m)!}{(n+m)!}}\,P_n^m(x).$$

Описание:

Функция P = legendre(n, X) вычисляет присоединенные функции Лежандра степени n и порядков m = 0, 1, ..., n для каждого элемента массива X. Аргумент n не должен превышать значения 256, а массив должен быть массивом действительных чисел из диапазона $-1 \le x \le 1$. Возвращаемый массив P имеет размерность на 1 большую размерности X, и каждый его элемент P(m+1,d1,d2...) содержит значения присоединенной функции Лежандра степени n и порядка m, вычисленные в точках X(d1,d2...).

Если X вектор и n равно двум, то матрица P имеет вид

$P_2^0(x(1))$	$P_2^0(x(2))$	$P_2^0(x(3))$	...
$P_2^1(x(1))$	$P_2^1(x(2))$	$P_2^1(x(3))$	...
$P_2^2(x(1))$	$P_2^2(x(2))$	$P_2^2(x(3))$	...

Функция S = legendre(n, X, 'sch') вычисляет полунормализованные по Шмидту присоединенные функции Лежандра.

Пример:

Оператор

```
P = legendre(2, 0.0 : 0.1 : 0.2)
P =
  -0.5000  -0.4850  -0.4400
        0  -0.2985  -0.5879
   3.0000   2.9700   2.8800
```

возвращает матрицу размера 3×3, каждая строка которой содержит значения присоединенной функции Лежандра степени 2 и соответствующего порядка 0, 1 или 2.

Построим графики соответствующих функций Лежандра, используя большее количество точек.

```
x = -1 : 0.01 : 1;
P = legendre(2, x);
plot(x, P(1, :)), grid, hold on
plot(x, P(2, :)), plot(x,P(3, :))
gtext('P20'), gtext('P21'), gtext('P22')
```

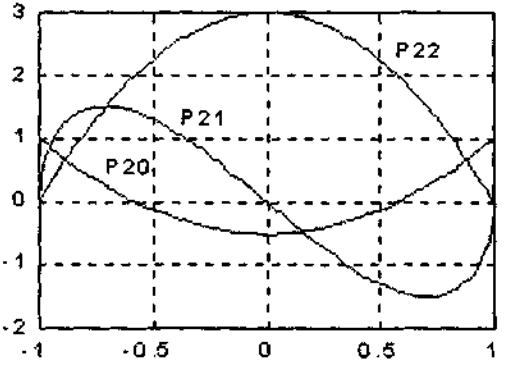


Рис. 6.14

В случае трехмерного массива X

```
X = rand(2, 4, 5);
P = legendre(2, X);
size(P)
ans =     3    2    4    5
```

результирующий четырехмерный массив имеет размер 3×2×4×5.

Сопутствующие функции: HTML-справка.

Ссылки:

1. Справочник по специальным функциям с формулами, графиками и таблицами/ Под ред. М. Абрамовича и И. Стиган: Пер. с англ. М.: Наука, 1979. 832 с.

7. МАТРИЦЫ И ЛИНЕЙНАЯ АЛГЕБРА

Матрица как математический объект возникает при решении конкретных вычислительных задач, и в первую очередь при решении систем линейных алгебраических уравнений и задач на собственные значения. Матрица в виде прямоугольной таблицы чисел очень схожа с массивом, однако прикладные задачи, которые порождают матрицы, определяют для них специальную совокупность допустимых операций, среди которых особое место занимает операция умножения. Для простейшего случая, когда умножается вектор-строка на вектор-столбец, такой операцией является операция скалярного произведения.

Матрицы широко используются при решении обыкновенных дифференциальных уравнений (ОДУ) и уравнений в частных производных, решении оптимальных задач и т. п.

Алгебраические задачи, связанные с матрицами, объединяются в раздел математики, получивший название *линейной алгебры*, который включает такие базисные задачи, как обращение и псевдообращение матриц, спектральное и сингулярное разложение матриц.

В вычислительном плане раздел линейной алгебры поддержан пакетами прикладных программ LINPACK, EISPACK, разработанными в 60-70-е годы ведущими специалистами, к числу которых принадлежит и основатель фирмы The MathWorks, Inc. Моулер (C. Moler). Изначальное назначение системы MATLAB состояло именно в том, чтобы создать диалоговую среду для работы с пакетами программ линейной алгебры.

Несмотря на кажущуюся завершенность, этот раздел развивается и в настоящее время в направлении создания новых операций: для работы с парами матриц (приведение пары матриц к форме Шура, рекуррентное сингулярное разложение пары прямоугольных матриц), решения матричных полиномов и полиномиальных матричных уравнений.

Рассмотрим функции системы MATLAB, которые поддерживают работу с матрицами, в следующей последовательности: операции с матрицами как числовыми массивами, коллекция тестовых матриц, характеристики матриц, решение систем линейных уравнений, вычисление собственных значений и сингулярных чисел, вычисление функций от матриц, работа с алгебраическими полиномами.

Операции над матрицами как числовыми массивами

EYE — **Формирование единичной матрицы**

Синтаксис:

```
Y = eye(n)
Y = eye(m, n)
Y = eye(size(A))
```

Описание:

Функция Y = ones(n) формирует единичную матрицу размера n×n.

Функция Y = ones(m, n) формирует единичную матрицу размера m×n.

Функция Y = ones(size(A)) формирует единичную матрицу, соразмерную с матрицей A.

Сопутствующие функции: ONES, RAND, RANDN, ZEROS.

DIAG **Формирование или извлечение диагоналей матрицы**

Синтаксис:

X = diag(v) v = diag(X)
X = diag(v, k) v = diag(X, k)

Описание:

Функция X = diag(v) формирует квадратную матрицу X с вектором v на главной диагонали.

Функция X = diag(v, k) формирует квадратную матрицу X порядка length(v)+abs(k) с вектором v на k-й диагонали.

Функция v = diag(X) извлекает из матрицы X главную диагональ.

Функция v = diag(X, k) извлекает из матрицы X диагональ с номером k; при k > 0 это номер k-й верхней диагонали, при k < 0 это номер k-й нижней диагонали.

Примеры:

diag(diag(X)) - диагональная матрица;
sum(diag(X)) - след матрицы X.
Оператор
diag(-m : m) + diag(ones(2*m, 1), 1) + diag(ones(2*m, 1), -1)
формирует трехдиагональную матрицу размера 2*m + 1.
Для m = 3 результирующая матрица имеет вид:

$$\begin{bmatrix} -3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 \end{bmatrix}$$

Сопутствующие функции: SPDIAGS, TRIL, TRIU.

TRIL **Формирование нижней треугольной матрицы (массива)**

Синтаксис:

L = tril(X)
L = tril(X, k)

Описание:

Функция L = tril(X) сохраняет нижнюю треугольную часть матрицы X.

Функция L = tril(X, k) сохраняет нижнюю треугольную часть матрицы X начиная с диагонали с номером k. При k > 0 это номер k-й верхней диагонали, при k < 0 это номер k-й нижней диагонали.

Пример:

Если $X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, то $\mathrm{tril}(X) = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 5 & 0 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, а $\mathrm{tril}(X, -2) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 4 & 5 & 0 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$.

Сопутствующие функции: DIAG, TRIU.

TRIU **Формирование верхней треугольной матрицы (массива)**

Синтаксис:

U = triu(X)

U = triu(X, k)

Описание:

Функция U = triu(X) сохраняет верхнюю треугольную часть матрицы (массива) X.

Функция U = triu(X, k) сохраняет верхнюю треугольную часть матрицы (массива) X начиная с диагонали с номером k. При k > 0 это номер k-й верхней диагонали, при k < 0 это номер k-й нижней диагонали.

Пример:

Если $X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, то $\mathrm{triu}(X) = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, а $\mathrm{triu}(X, -2) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \\ 0 & 5 & 6 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix}$.

Сопутствующие функции: DIAG, TRIL.

FLIPLR **Отражение матрицы относительно вертикальной оси**

Синтаксис:

B = fliplr(A)

Описание:

Функция B = fliplr(A) переставляет столбцы матрицы A симметрично относительно вертикальной оси. Если матрица A имеет нечетное число столбцов, то средний столбец остается на своем месте.

Пример:

Если $A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$, то fliplr(A) = $\begin{bmatrix} 4 & 3 & 2 & 1 \\ 4 & 3 & 2 & 1 \\ 4 & 3 & 2 & 1 \end{bmatrix}$.

Сопутствующие функции: FLIPUD, ROT90.

FLIPUD **Отражение матрицы относительно горизонтальной оси**

Синтаксис:

B = flipud(A)

Описание:

Функция B = flipud(A) переставляет строки матрицы A симметрично относительно горизонтальной оси. Если матрица A имеет нечетное число строк, то средняя строка остается на своем месте.

Пример:

Если $A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \end{bmatrix}$, то flipud(A) = $\begin{bmatrix} 4 & 4 & 4 \\ 3 & 3 & 3 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$.

Сопутствующие функции: FLIPLR, ROT90.

ROT90 **Поворот матрицы на 90 градусов**

Синтаксис:

B = rot90(A)
B = rot90(A, k)

Описание:

Функция B = rot90(A) осуществляет поворот матрицы A размером m×n на 90 градусов против часовой стрелки.

Функция B = rot90(A, k) осуществляет поворот матрицы A размером m×n на 90 * k, где k = ±1, ±2,...

Пример:

Пусть $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$.

Тогда B = rot90(A) = $\begin{bmatrix} 3 & 6 \\ 2 & 5 \\ 1 & 4 \end{bmatrix}$, а C = rot90(a, -2) = $\begin{bmatrix} 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$.

Сопутствующие функции: VIEW, FLIPUD, FLIPLR.

Коллекция тестовых матриц

COMPAN **Сопровождающая матрица для многочлена**

Синтаксис:

C = compan(p)

Описание:

Функция C = compan(p) формирует сопровождающую матрицу C для многочлена, заданного своими коэффициентами в виде вектора p. Первая строка этой матрицы есть вектор -p(2 : n)/p(1).

Определение сопровождающей матрицы и ее применение на практике описано в [1-3].

Примеры:

Полиному (x-1)(x-2)(x-3) = x3 - 7x + 6 соответствует вектор p = [1 0 -7 6], для которого сопровождающая матрица имеет вид:

```
C = compan(p)
C =
     0    7   -6
     1    0    0
     0    1    0
```

Замечание:

Эта функция изменена по сравнению с версией MATLAB 4.

Сопутствующие функции: POLY, POLYVAL.

Ссылки:

1. Wilkinson J. H. *The Algebraic Eigenvalue Problem.* Oxford, 1965. - То же: Уилкинсон Дж. Х. Алгебраическая проблема собственных значений. М.: Наука, 1970.
2. Golub G. H., Van Loan C. F. *Matrix Computations.* Baltimore; Maryland: Johns Hopkins University Press,1989.
3. Kenney C., Laub A. J. Controllability and stability radii for companion form systems.//*Math. Control Signals Systems.* 1988. Vol. 1. P. 239-256.

HADAMARD **Матрица Адамара (Hadamard matrix)**

Синтаксис:

H = hadamard(n)

Описание:

Функция H = hadamard(n) возвращает матрицу Адамара порядка n.

Матрицы Адамара встречаются в различных приложениях - комбинаторном и численном анализе, обработке сигналов [1, 2]. Это матрицы, составленные из 1 и -1, столбцы которых ортогональны, так что справедливо соотношение

H' * H = n * I,

где [n, n] = size(H) и I = eye(n, n).

Матрица Адамара порядка n > 2 существует только тогда, когда n кратно 4. Данный алгоритм вычисляет матрицы Адамара для тех случаев, когда величины n, n/12, n/20 являются степенями по основанию 2 [3].

Пример:

```
H = hadamard(8)
H =
  1   1   1   1   1   1   1   1
  1  -1   1  -1   1  -1   1  -1
  1   1  -1  -1   1   1  -1  -1
  1  -1  -1   1   1  -1  -1   1
  1   1   1   1  -1  -1  -1  -1
  1  -1   1  -1  -1   1  -1   1
  1   1  -1  -1  -1  -1   1   1
  1  -1  -1   1  -1   1   1  -1
```

Картина линий уровня для этой матрицы напоминает ковер

contour(hadamard(8))

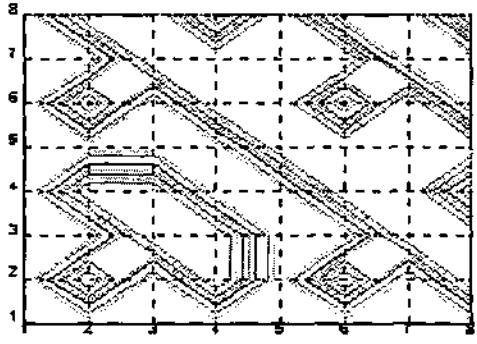


Рис. 7.1

Сопутствующие функции: COMPAN, HANKEL, TOEPLITZ.

Ссылки:

1. Ryser H. J. *Combinatorial Mathematics*. New York: John Wiley&Sons, 1963.
2. Pratt W. K. *Digital Signal Processing*. New York: John Wiley&Sons, 1978.
3. Golomb S. W., Baumert L. D. The search for Hadamard matrices//*Amer. Math. Monthly*. Vol. 70, 1963. P. 12-17.

HANKEL — Матрица Ганкеля (Hankel matrix)

Синтаксис:

```
H = hankel(c)
H = hankel(c, r)
```

Определение:

Матрица Ганкеля - это симметрическая матрица с постоянными значениями на антидиагоналях с элементами h(i, j) = p(i+j–1), где вектор p = [c r(2:end)] полностью определяет задание матрицы.

Описание:

Функция H = hankel(c) возвращает квадратную матрицу Ганкеля, первый столбец которой совпадает с вектором c, а все элементы H(i, j), i + j > n, лежащие ниже второй главной диагонали, равны нулю.

Функция H = hankel(c, r) возвращает матрицу Ганкеля, первый столбец которой совпадает с вектором c, а последняя строка - с вектором r. Если последний элемент вектора c не равен первому элементу вектора r, то возникает конфликт на второй главной диагонали, когда предпочтение отдается элементу вектора c.

Примеры:

```
c = [1 2 3]; H = hankel(c)
H =
        1    2    3
        2    3    0
        3    0    0
c = 1:3; r = 7:10; H = hankel(c, r)
Column wins anti-diagonal conflict.
```

Столбец выигрывает конфликт на второй главной диагонали

```
H =
        1    2    3    8
        2    3    8    9
        3    8    9    10
p = [c r(2:end)]
p = 1    2    3    8    9   10
```

Сопутствующие функции: HADAMARD, TOEPLITZ.

HILB, INVHILB **Матрица Гильберта (Hilbert matrix)**

Синтаксис:

```
H = hilb(n)
H = invhilb(n)
```

Описание:

Функция H = hilb(n) формирует матрицу Гильберта порядка n. Элементы этой матрицы определяются следующим образом:

$$H(i, j) = \frac{1}{i + j - 1}.$$

Матрица Гильберта - это пример очень плохо обусловленной по отношению к операции обращения матрицы [1].

Функция H = invhilb(n) формирует матрицу, обратную матрице Гильберта порядка n. Точная обратная матрица - это матрица, элементами которой являются целые числа. Точное представление такой матрицы в арифметике с плавающей точкой возможно только тогда, когда порядок матрицы не превышает 13. Для больших значений n функция invhilb(n) формирует только приближенную матрицу.

Сравнение функций invhilb(n) и inv(hilb(n)) позволяет выявить несколько источников ошибок:

- ошибки, вызванные функцией hilb(n);
- ошибки, связанные с процедурой обращения;
- ошибки, вызванные функцией invhilb(n).

Оказывается, что первый источник ошибок, связанный с представлением правильных дробей вида 1/3 или 1/5 в арифметике с плавающей точкой, наиболее существенный.

Пример:

Матрица Гильберта порядка 4 имеет число обусловленности 1.5514e+004.
Ее обратная матрица - это целочисленная матрица вида

```
invhilb(4)
    16   -120    240   -140
  -120   1200  -2700   1680
   240  -2700   6480  -4200
  -140   1680  -4200   2800
```

а результат обращения в арифметике с плавающей точкой

```
inv(hilb(4))
    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
```

Сопутствующие функции: HTML-справка.

Ссылки:

1. Forsythe G. E., Moler C. B. *Computer Solution of Linear Algebraic Systems.* Prentice-Hall N. Y., 1967.

MAGIC **Магический квадрат**

Синтаксис:

M = magic(n)

Описание:

Функция M = magic(n) для n > 3 формирует специальную квадратную матрицу порядка n, элементами которой являются целые числа от 1 до n^2, суммы элементов которой по строкам и столбцам равны. Эта функция

магического квадрата была включена в состав системы MATLAB в 1993 году и подробно описана в работе [1]. Сумму элементов по строкам (столбцам) назовем *инвариантом магического квадрата* и обозначим μ_n. Значение инварианта зависит от n и равно

$\mu_n = n(n^2+1)/2$.

Если матрицу магического квадрата отнормировать делением на ее инвариант, то получим дважды стохастическую матрицу, обладающую тем свойством, что ее матричная норма любого порядка равна 1 [2].

Ранг матрицы M зависит от n следующим образом [2]:

$$\text{rank(magic(n))} = \begin{cases} n & \text{при} \quad n = 2k+1, \quad k = 1, 2, \ldots \\ n/2+2 & \text{при} \quad n/2 = 2k-1, \quad k = 1, 2, \ldots \\ 3 & \text{при} \quad n = 4k, \quad k = 1, 2, \ldots \end{cases}$$

График функции rank(magic(n)) для 3 <= n <= 32 показан на рис. 7.2.

```
for i=3:32
   r(i) = rank(magic(i));
end
 plot(1:32, r, '-o')
```

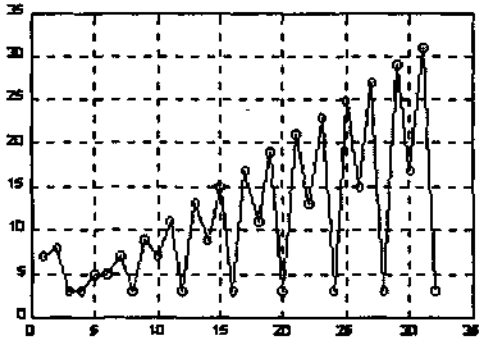


Рис. 7.2

Сопутствующие функции: RAND, ONES.

Ссылки:

1. Moler C. B. MATLAB's magical mystery tour//*The MathWorks Newsletter*. 1993. Vol. 7(1).
2. Higham N. J. The Test Matrix Toolbox for MATLAB (version 3.0)//*Numerical Analysis Report*. Manchester, 1995. Vol. 276.

PASCAL **Матрица Паскаля (Pascal matrix)**

Синтаксис:

```
P = pascal(n)
P = pascal(n, k)
```

Описание:

Функция P = pascal(n) формирует симметрическую положительно определенную квадратную матрицу порядка n, которая составлена из элементов треугольника Паскаля. Треугольник Паскаля представляет собой коэффициенты разложения бинома $(1 + w)^i$, записанные в следующем виде:

$$1$$
$$1 + w$$
$$1 + 2w + w^2$$
$$1 + 3w + 3w^2 + w^3$$
$$1 + 4w + 6w^2 + 4w^3 + w^4$$

а матрицы Паскаля порядка 3 и 4 имеют следующий вид:

```
pascal(3) =                    pascal(4) =
   1   1   1                      1   1   1   1
   1   2   3                      1   2   3   4
   1   3   6                      1   3   6  10
                                  1   4  10  20
```

Функция pascal(n, 0) равносильна функции pascal(n).

Матрица P = pascal(n, 1) - это нижняя треугольная матрица в разложении Холецкого для матрицы pascal(n) с точностью до знаков чисел в столбцах. Эта матрица обладает свойством P2 = I, где I - единичная матрица [1].

Матрица P = pascal(n, 2) - это матрица, полученная в результате транспонирования и перестановок в матрице pascal(n, 1). Эта матрица обладает свойством $P^3 = I$, где I - единичная матрица [2, 3].

Пример:

Сформируем матрицы pascal(3), pascal(3, 1) и pascal(3, 2):

```
P = pascal(3)        P1=pascal(3,1)        P2=pascal(3,2)
P =                  P1 =                  P2 =
   1   1   1            1   0   0             0   0  -1
   1   2   3            1  -1   0             0  -1   2
   1   3   6            1  -2   1            -1  -1   1
                        P1^2                  P2^3
                        ans =                 ans =
                           1   0   0             1   0   0
                           0   1   0             0   1   0
                           0   0   1             0   0   1.
```

Сопутствующие функции: CHOL.

Ссылки:

1. Higham N. J. The Test Matrix Toolbox for MATLAB (version 3.0)//Vol. 276. *Numerical Analysis Report.* Manchester, 1995.
2. Higham N. J. Accuracy and Stability of Numerical Algorithms. *Society for Industrial and Applied Mathematics.* Philadelphia, 1996.
3. Brawer R., Pirovino M. The linear algebra of the Pascal matrix//Vol. 174. *Linear Algebra and Appl.* 1992. P. 13-23.

ROSSER **Матрица Рессера (Rosser matrix)**

Синтаксис:

R = rosser

Описание:

Функция R = rosser формирует тестовую матрицу для классической симметрической проблемы собственных значений. Эта матрица служила камнем преткновения для многих алгоритмов вычисления собственных значений. Только QR-алгоритм Франсиса, усовершенствованный Уилкинсоном [1] и реализованный в пакете программ EISPACK и в системе MATLAB, позволяет справиться с указанной проблемой.

Матрица rosser - это матрица порядка 8 с целочисленными элементами, она обладает следующим спектром собственных значений:

- пара кратных значений;
- 3 близких собственных значения;
- нулевое собственное значение;
- малое ненулевое собственное значение.

Матрица Рессера

```
R =
     611     196    -192     407      -8     -52     -49      29
     196     899     113    -192     -71     -43      -8     -44
    -192     113     899     196      61      49       8      52
     407    -192     196     611       8      44      59     -23
      -8     -71      61       8     411    -599     208     208
     -52     -43      49      44    -599     411     208     208
     -49      -8       8      59     208     208      99    -911
      29     -44      52     -23     208     208    -911      99
```

Сравним собственные значения матрицы Рессера, вычисленные с помощью функции eig(rosser) с точными значениями.

eig(rosser)	*Точные значения*	*Вычисленные точные значения*
1020.04901843	10*(1 + sqrt(10201))	1020
1020	1020	1020
1019.90195135928	510 + 100*sqrt(26)	1019.90195135928
1000	1000	1000
1000	1000	1000
0.0980486407217549	510 - 100*sqrt(26)	0.0980486407215722
8.11989364635224e-013	0	0
-1020.04901843	-10*(1 + sqrt(10201))	-1020

Сопутствующие функции: EIG, WILKINSON.

Ссылки:

1. Уилкинсон, Райнш. *Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра:* Пер. с англ. М.: Машиностроение, 1976. 390 с.

TOEPLITZ **Матрица Теплица (Toeplitz matrix)**

Синтаксис:

```
T = toeplitz(c)
T = toeplitz(c, r)
```

Описание:

Функция T = toeplitz(c) возвращает симметрическую матрицу Теплица, определяемую однозначно вектором c.

Функция T = toeplitz(c, r) возвращает несимметрическую матрицу Теплица, первый столбец которой совпадает с вектором c, а первая строка - с вектором r. Если первый элемент вектора c не равен первому элементу вектора r, то возникает конфликт на главной диагонали, когда предпочтение отдается элементу вектора c.

Примеры:

```
c = 1 : 4; T = toeplitz(c)
T =
   1   2   3   4
   2   1   2   3
   3   2   1   2
   4   3   2   1
c = 1 : 4; r = 1.5 : 4.5; T = toeplitz(c, r)
Column wins diagonal conflict.
```

Столбец выигрывает конфликт на главной диагонали

```
T =
        1.0000  2.5000  3.5000  4.5000
        2.0000  1.0000  2.5000  3.5000
        3.0000  2.0000  1.0000  2.5000
        4.0000  3.0000  2.0000  1.0000
```

Сопутствующие функции: HANKEL.

VANDER **Матрица Вандермонда (Vandermonde matrix)**

Синтаксис:

```
V = vander(x)
```

Описание:

Функция V = vander(x) возвращает матрицу Вандермонда порядка length(x), j-й столбец которой определяется соотношением V(:, j) = x^(n - j). Второй справа столбец совпадает с вектором x.

Примеры:

```
x =        1      2      3      4
V =vander(x)
V =
           1      1      1      1
           8      4      2      1
          27      9      3      1
          64     16      4      1
```

Сопутствующие функции: HTML-справка.

WILKINSON **Матрица Уилкинсона (Wilkinson's matrix)**

Синтаксис:

W = wilkinson(n)

Описание:

Функция W = wilkinson(n) формирует тестовую матрицу Уилкинсона для задачи на собственные значения. Это симметрическая трехдиагональная матрица, наибольшие собственные значения которой попарно близки, но не являются кратными.

Матрица Уилкинсона порядка 7 имеет следующий вид:

```
W = wilkinson(7)
W =
     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
```

Обычно в качестве тестовой используется матрица 21-го порядка, имеющая следующие собственные значения:

```
eig(wilkinson(21))
        1.074619418290332e+001
        1.074619418290340e+001
        9.210678647304920e+000
        9.210678647361334e+000
        8.038941115814277e+000
        8.038941122829026e+000
        7.003951798616379e+000
        7.003952209528680e+000
        6.000234031584172e+000
        6.000217522257103e+000
        5.000244425001918e+000
        4.999782477742906e+000
        4.004354023440860e+000
        3.996048201383629e+000
        3.043099292578829e+000
        2.961058884185733e+000
        2.130209219362509e+000
        1.789321352695084e+000
        9.475343675292956e-001
        2.538058170966829e-001
       -1.125441522119983e+000
```

Цифры, являющиеся неточными, выделены жирным шрифтом.

Сопутствующие функции: EIG, GALLERY, PASCAL.

GALLERY **Набор тестовых матриц**

Синтаксис:

[out1, out2, ...] = gallery('<*имя_матрицы*>', param1, param2, ...)
A = gallery(3)
A = gallery(5)

Описание:

Функция [out1, out2, ...] = gallery('<*имя_матрицы*>', param1, param2, ...) возвращает тестовую матрицу с именем <*имя_матрицы*> из набора приведенных ниже тестовых матриц. При вызове, как правило, указывается в качестве входного параметра порядок матрицы, но могут потребоваться и другие параметры.

Набор включает около 50 матриц, используемых для тестирования алгоритмов и других целей:

cauchy	cycol	hanowa	krylov	orthog	redheff	wilk
chebspec	dorr	house	lauchli	parter	riemann	
chebvand	dramadah	invhess	lehmer lesp	pei poisson	ris	
chow	fiedler	invol			smoke	
circul	forsythe	ipjfact	lotkin	prolate	toeppd	
clement	frank	jordbloc	minij	rando	tridiag triw	
compar	gearmat	kahan	moler	randhess	wathen	
condex	grcar	kms	neumann	randsvd		

Функция A = gallery(3) возвращает плохо обусловленную по отношению к задаче обращения тестовую матрицу порядка 3 с числом обусловленности 2.76e+005.

Функция A = gallery(5) возвращает целочисленную тестовую матрицу порядка 5 для полной проблемы собственных значений.

cauchy **Матрица Коши**

Синтаксис:

C = gallery('cauchy', x, y)
C = gallery('cauchy', x)

Описание:

Функция C = gallery('cauchy', x, y) возвращает матрицу размера n×n, элементы которой равны C(i, j) = 1/(x(i)+y(j)), где x и y - векторы длины n. Если указаны в качестве x и y скаляры, то они интерпретируются как векторы 1:x и 1:y.

Функция C = gallery('cauchy', x) также возвращает матрицу Коши размера n×n, элементы которой равны C(i, j) = 1/(x(i)+x(j)).

Известны точные выражения для обратной матрицы и определителя матрицы Коши. Определитель det(C) не равен 0, если векторы x и y имеют различные элементы. Матрица Коши является положительно определенной, если выполняются условия 0 < x(1) <... < x(n) и 0 < y(1) < ... < y(n).

chebspec **Матрица Чебышева**

Синтаксис:

C = gallery('chebspec', n, <*переключатель*>)

Описание:

Функция C = gallery('chebspec', n, <*переключатель*>) возвращает матрицу Чебышева спектрального дифференцирования размера n×n. Переключатель может принимать значения 0 или 1; по умолчанию 0. Для значения 0 матрица C нильпотентна ($C^n = 0$), имеет нуль-вектор ones(n, 1) и подобна блоку Жордана с нулевым собственным значением. Для значения 1 матрица C невырожденная, хорошо обусловленная матрица, собственные значения которой имеют отрицательные действительные части.

chebvand **Матрица Вандермонда для полиномов Чебышева**

Синтаксис:

C = gallery('chebvand', p)

C = gallery('chebvand', m, p)

Описание:

Функция C = gallery('chebvand', p) возвращает матрицу Вандермонда для полиномов Чебышева, заданных на точках, определяемых вектором p.

Функция C = gallery('chebvand', m, p) возвращает прямоугольную матрицу с m строками.

Если p вектор, то $C(i, j) = T_{i-1}(p(j))$, где T_{i-1} полином Чебышева степени i-1; если в качестве p указан скаляр, то он интерпретируется как вектор p равноудаленных точек на интервале [0,1], используемых для вычисления C.

chow **Сингулярная матрица Теплица в нижней форме Хессенберга**

Синтаксис:

C = gallery('chow', n, alpha, delta)

Описание:

Функция C = gallery('chow', n, alpha, delta) возвращает матрицу C вида H(alpha) + delta*eye(n), где $H_{i,j}(\alpha) = \alpha^{i-j+1}$. Здесь n - порядок матрицы, а параметры alpha и delta - скаляры; по умолчанию они равны соответственно 1 и 0.

Матрица H(alpha) имеет p = floor(n/2) собственных значений, равных 0; остальные равны 4*alpha*cos(k*pi/(n+2))^2, k = 1 : n-p.

Пример:

```
C = gallery('chow', 3)
C =
   1   1   0
   1   1   1
   1   1   1
```

```
eig(C)
ans =
   0.3820
   2.6180
        0
n = length( C )
n =  3
p = floor(n/2)
p =  1
for k = 1:2
  d(k) = 4*cos(k*pi/(n+2))^2;
end
d
d =  2.6180    0.3820
```

circul **Циркулянтная матрица**

Синтаксис:

C = gallery('circul', v)

Описание:

Функция C = gallery('circul', v) возвращает циркулянтную матрицу C, первая строка совпадает с вектором v. Каждая следующая строка получена циклической перестановкой элементов первой строки.

Если в качестве v указан скаляр k, то он интерпретируется как вектор, то есть C = gallery('circul', 1:k).

Для циркулянтной матрицы порядка n система собственных значений и собственных векторов известна точно: если t - корень n-й степени из 1, то собственное значение определяется как скалярное произведение векторов v и $w = [1\ t\ t^2 \dots t^{n-1}]$, а собственный вектор для соответствующего собственного значения равен w^H.

Пример:

```
C = gallery('circul', 3)
C =
    1    2    3
    3    1    2
    2    3    1
[R, D] = eig(C)
R =
  0.5774          0.2562 + 0.5174i  0.2562 - 0.5174i
  0.5774          0.3199 - 0.4806i  0.3199 + 0.4806i
  0.5774         -0.5762 - 0.0368i -0.5762 + 0.0368i
D =
  6.0000              0                 0
       0        -1.5000 + 0.8660i        0
       0              0          -1.5000 - 0.8660i
```

Введем нормировку матрицы собственных векторов

```
for i=1:3
   RN(:, i)=R(:, i)/R(1, i);
end
RN
RN =
  1.0000          1.0000                1.0000
  1.0000         -0.5000 - 0.8660i  -0.5000 + 0.8660i
  1.0000         -0.5000 + 0.8660i  -0.5000 - 0.8660i
```

Нормированные собственные векторы совпадают с векторами w^H.

clement **Трехдиагональная матрица с нулевой диагональю**

Синтаксис:

```
C = gallery('clement', n, sym)
```

Описание:

Функция C = gallery('clement', n, sym) возвращает трехдиагональную матрицу C порядка n с нулями на диагонали и известными собственными значениями. Матрица C вырожденна, если n нечетно; для невырожденной матрицы ее обратная имеет около 64 % нулевых элементов. Собственные значения для четных n принадлежат последовательности {n-1, n-3, n-5 ...} со знаками плюс и минус; для нечетных n добавляется собственное значение 0 или 1. Входной параметр sym определяет, является ли матрица Клемента симметрической; по умолчанию sym равно 0, что соответствует несимметрической матрице; для значения sym, равного 1, матрица Клемента симметрическая.

Пример:

```
C = gallery('clement', 4)
C =
   0   1   0   0
   3   0   2   0
   0   2   0   3
   0   0   1   0
eig(C)
ans =
   3.0000
  -3.0000
   1.0000
  -1.0000
```

compar **Матрицы сравнения**

Синтаксис:

```
C = gallery('compar', A)
C = gallery('compar', A, 1)
```

Описание:

Функции C = gallery('compar', A) и C = gallery('compar', A, 0) возвращают матрицу C = diag(B) - tril(B, -1) - triu(B, 1), где B = abs(A).

Функция C = gallery('compar', A, 1) возвращает матрицу C, диагональные элементы которой равны абсолютным значениям диагональных элементов матрицы A, а внедиагональные элементы каждой строки заменяются максимальным по модулю значением элемента этой строки со знаком минус. Матрицы в треугольной форме остаются без изменения.

Пример:

```
A =  gallery('chebspec',  4)
A =
   3.1667  -4.0000   1.3333  -0.5000
   1.0000  -0.3333  -1.0000   0.3333
  -0.3333   1.0000   0.3333  -1.0000
   0.5000  -1.3333   4.0000  -3.1667
C = gallery('compar', A)
C =
   3.1667  -4.0000  -1.3333  -0.5000
  -1.0000   0.3333  -1.0000  -0.3333
  -0.3333  -1.0000   0.3333  -1.0000
  -0.5000  -1.3333  -4.0000   3.1667

C = gallery('compar', A, 1)
C =
   3.1667  -4.0000  -4.0000  -4.0000
  -1.0000   0.3333  -1.0000  -1.0000
  -1.0000  -1.0000   0.3333  -1.0000
  -4.0000  -4.0000  -4.0000   3.1667
```

condex — Контрпримеры матриц при оценке чисел обусловленности

Синтаксис:

C = gallery('condex', n, k, theta)

Описание:

Функция C = gallery('condex', n, k, theta) возвращает для различных значений параметра k следующие контрпримеры матриц, связанные с оценкой чисел обусловленности различными алгоритмами:

k	*n*	*Алгоритм оценки*
1	4	LINPACK (rcond)
2	3	LINPACK (rcond)
3	любое	LINPACK (rcond)
4	≥ 4	SONEST (Higham 1988)

Если для соответствующих значений порядок превышает указанный, то матрица расширяется до требуемого порядка единичной; если порядок меньше, то формируется требуемая матрица.

Пример:

```
C = gallery('condex', 4, 1)
C =
   1   -1  -200    0
   0    1   100  -100
   0    1   101  -101
   0    0     0   100
```

1/rcond(C)	condest(C)	condeig(C)	eig(C)
565.59	40100	2.6687	1
		1.9612	0.0098049
		123.72	101.99
		125.37	100

```
C = gallery('condex', 3, 2)
C =
      1      0.9998    -2
      0      0.01      -0.01
      0      0          1
```

1/rcond(C)	condest(C)	condeig(C)	eig(C)
7.4884	601.94	5.0005e+015	1
		1.4213	0.01
		5.0005e+015	1

```
C = gallery('condex', 4, 3)
C =
   1   0   0   0
  -1   1   0   0
  -1  -1   1   0
  -1  -1  -1  -1
```

1/rcond(C)	condest(C)	condeig(C)	eig(C)
4	32	1.3815	-1
		2.0882e+015	1
		1.7469e+030	1
		1.7469e+030	1

```
C = gallery('condex', 4, 4)
C =
        1          -5.8287e-014  5.5511e-015  -1.8735e-014
 -5.8287e-014       59.738       -10.68        -48.058
  5.5511e-015      -10.68         2.9417        8.7379
 -1.8735e-014      -48.058        8.7379       40.32
```

1/rcond(C)	condest(C)	condeig(C)	eig(C)
118.48	138.98	1	1
		1	1
		1	1
		1	101

cycol **Матрица с периодически повторяющимися столбцами**

Синтаксис:

C = gallery('cycol', n, k)
C = gallery('cycol', [m n], k)

Описание:

Функция C = gallery('cycol', n, k) возвращает матрицу порядка n с периодически повторяющимися столбцами; период повторения задается параметром k (по умолчанию k = round(n/4)). Повторяющиеся столбцы формируются как массив случайных чисел randn(m, k). Ранг такой матрицы не превышает значения k.

Функция C = gallery('cycol', [m n], k) возвращает прямоугольную матрицу порядка m×n с периодически повторяющимися столбцами.

dorr **Матрица Дорра**

Синтаксис:

D = gallery('dorr', n, theta)
[c, d, e] = gallery('dorr', n, theta)

Описание:

Функция D = gallery('dorr', n, theta) возвращает плохо обусловленную трехдиагональную разреженную матрицу порядка n с диагонально преобладающими элементами. Число обусловленности зависит от параметра theta; для малых неотрицательных theta матрица Дорра плохо обусловлена (по умолчанию theta = 0.01).

Функция [c, d, e] = gallery('dorr', n, theta) возвращает диагонали матрицы Дорра. Для формирования самой матрицы Дорра надо использовать оператор D = gallery('tridiag', c, d, e).

dramadah **Анти-адамаровы матрицы**

Синтаксис:

D = gallery('dramadah', n, k)

Описание:

Функция D = gallery('dramadah', n, k) возвращает матрицу порядка n из нулей и единиц, которая имеет достаточно большое значение параметра mu(D) = norm(inv(D), 'fro'), хотя и не обязательно максимальное. Матрица называется анти-адамаровой, если значение параметра mu(D) максимально. Параметр определяет тип формируемой матрицы:

- k = 1 (по умолчанию). D - матрица Теплица с определителем abs(det(A)) = 1 и mu(A) > c(1.75)^n, где константа c приблизительно оценивается значением 0.43;
- k = 2. D - верхняя треугольная матрица Теплица с кратными собственными значениями, равными 1;

- k = 3. D - матрица Теплица с максимальным определителем, равным n-му члену последовательности чисел Фибоначчи, среди всех матриц в нижней форме Хессенберга. Определенный интерес может представить распределение собственных значений.

Примеры:

```
D = gallery('dramadah', 10, 1);
mu = norm(inv(D), 'fro')
mu =   109.65

D = gallery('dramadah', 10, 2);
mu = norm(inv(D), 'fro')
mu =    55.091

D = gallery('dramadah', 10, 3);
mu = norm(inv(D), 'fro')
mu =   2.4421
det(D)
ans =   55
```

Это 10-й член последовательности Фибоначчи $u_n + u_{n+1} = u_{n+2}$, (u_0=0, u_1=1, n=0, 1, 2 ...).

```
see(D, -1)
```

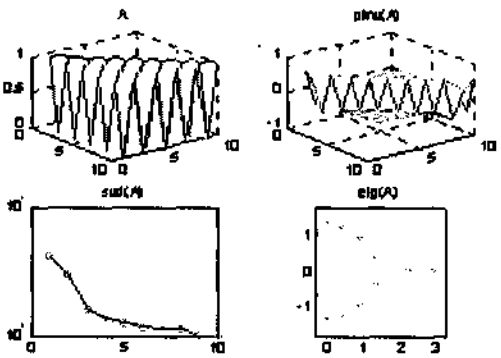


Рис. 7.3

fiedler **Матрица Фидлера**

Синтаксис:

```
F = gallery('fiedler', c)
```

Описание:

Функция F = gallery('fiedler', c), где c - вектор длины n, возвращает симметрическую матрицу порядка n с элементами abs(c(i) - c(j)). Если в качестве c указан скаляр k, то он интерпретируется как вектор 1:k.

Обратная матрица является трехдиагональной с дополнительными элементами в позициях (1, n) и (n, 1). Матрица Фидлера имеет преобладающее положительное собственное значение, и остальные собственные значения отрицательны.

Точные соотношения для inv(A) и det(A) приведены в [1, с.159] и принадлежат Фидлеру (Fiedler).

Примеры:

```
F = gallery('fiedler',1:4),
F =
   0   1   2   3
   1   0   1   2
   2   1   0   1
   3   2   1   0
det(F)
ans = -12
inv(F)
ans =
     -0.3333   0.5000        0    0.1667
      0.5000  -1.0000   0.5000         0
           0   0.5000  -1.0000    0.5000
      0.1667   0.0000   0.5000   -0.3333
eig(F)
ans =
   -0.58579
    -1.1623
    -3.4142
     5.1623
```

Ссылки:

1. Todd J. Basic Numerical Mathematics. Vol. 2: Numerical Algebra. Basel; New York: Birkhauser: Academic Press, 1977.

forsythe — Матрица Форсайта

Синтаксис:

```
F = gallery('forsythe', n, alpha, lambda)
```

Описание:

Функция F = gallery('forsythe', n, alpha, lambda) возвращает матрицу порядка n, составленную из клетки Жордана с собственным значением lambda и дополнительным элементом F(n, 1) = alpha. По умолчанию значения alpha и lambda равны соответственно sqrt(eps) и 0.

Характеристический полином матрицы Форсайта равен det(F-s*I) = = (lambda - s)^n - alpha*(-1)^n.

Примеры:

```
F = gallery('forsythe', 4)
F =
         0         1         0         0
         0         0         1         0
         0         0         0         1
  1.4901e-008      0         0         0
eig(F)
ans =
  -0.0110
  -0.0000 + 0.0110i
  -0.0000 - 0.0110i
   0.0110
```

frank **Матрица Франка**

Синтаксис:

F = gallery('frank', n, k)

Описание:

Функция F = gallery('frank', n, k) возвращает матрицу Франка порядка n. Это матрица в верхней форме Хессенберга с определителем, равным 1. Параметр k по умолчанию равен 0; k = 1 элементы матрицы отражаются относительно антидиагонали.

Собственные значения этой матрицы могут быть выражены через нули полиномов Эрмита соответствующего порядка; все они положительны и образуют пары взаимно обратных чисел. Если порядок матрицы нечетный, то одно из собственных значений равно 1.

Матрица Франка имеет floor(n/2) плохо обусловленных минимальных собственных значений.

Примечание. Полиномы Эрмита определяются по следующей формуле:

$$e^{-\frac{x^2}{2}} H_n(x) = \frac{1}{n!}\frac{d^n}{dx^n} e^{-\frac{x^2}{2}}.$$

Примеры:

F = gallery('frank', 7)

F =							eig(F) =	condeig(F) =
7	6	5	4	3	2	1	16.003	1.4837
6	6	5	4	3	2	1	7.4676	1.5761
0	5	5	4	3	2	1	2.9992	1.5879
0	0	4	4	3	2	1	1	11.163
0	0	0	3	3	2	1	0.062487	252.66
0	0	0	0	2	2	1	0.13391	362.98
0	0	0	0	0	1	1	0.33342	120.99

gearmat **Матрица Гира**

Синтаксис:

G = gallery('gearmat', n, i, j)

Описание:

Функция G = gallery('gearmat', n, i, j) возвращает матрицу Гира порядка n с единицами на первой поддиагонали и первой наддиагонали, а также с элементом sign(i) в позиции (1, abs(i)) и элементом sign(j) в позиции (n, n+1 - abs(j)); остальные элементы матрицы равны 0. По умолчанию i = n, а j = -n.

Матрица Гира вырожденна, может иметь собственные значения кратности 2 или 3 и быть неполного ранга. Собственные значения этой матрицы равны 2*cos(a), а собственные векторы имеют вид [sin(w+a), sin(w+2a), ..., sin(w+na)]. Значения параметров a и w могут быть получены из [1].

Примеры:

G = gallery('gearmat', 8)

G								eig(G)	condeig(G)
0	1	0	0	0	0	0	1	-1.4142	1.4142
1	0	1	0	0	0	0	0	-5.2042e-018 +1.0793e-008i	9.2656e+007
0	1	0	1	0	0	0	0	-5.2042e-018 -1.0793e-008i	9.2656e+007
0	0	1	0	1	0	0	0	1.4142	1.4142
0	0	0	1	0	1	0	0	1.8478	1.0824
0	0	0	0	1	0	1	0	-1.8478	1.0824
0	0	0	0	0	1	0	1	0.76537	2.6131
-1	0	0	0	0	0	1	0	- 0.76537	2.6131

```
rank(G)
ans =   7
```

Ссылки:

1. Gear C. W. A Simple Set of Test Matrices for Eigenvalue Programs//Math. Comp. 1969. Vol. 23. P. 119–125.

grcar **Матрица Теплица с чувствительными собственными значениями**

Синтаксис:

G = gallery('grcar', n, k)

Описание:

Функция G = gallery('grcar', n, k) возвращает матрицу Теплица порядка n с единицами на главной диагонали и k наддиагоналях, а также со значениями -1 на первой поддиагонали. Значение k по умолчанию равно 3. Собственные значения обладают очень высокой чувствительностью.

Пример:

```
G = gallery('grcar', 11)
see(G, -1)
```

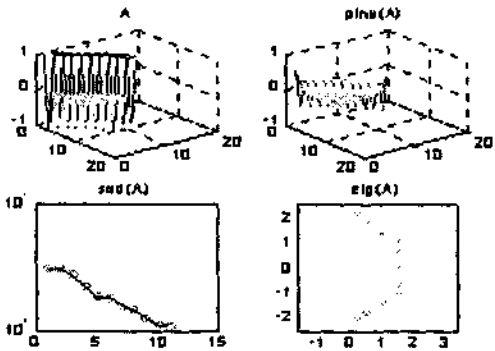


Рис. 7.4

hanowa

Матрица с собственными значениями с постоянной действительной частью

Синтаксис:

```
H= gallery('hanowa', n, d)
```

Описание:

Функция H= gallery('hanowa', n, d) возвращает матрицу порядка n, которая может быть представлена в следующем блочном виде:

[d*eye(m) –diag(1:m)
  diag(1:m) d*eye(m)].

Порядок n четный и равен 2m. Матрица имеет комплексные собственные значения d ± k*i, где 1 <= k <= m. По умолчанию d равно -1.

Пример:

```
H= gallery('hanowa', 6)
H =
   -1    0    0   -1    0    0
    0   -1    0    0   -2    0
    0    0   -1    0    0   -3
    1    0    0   -1    0    0
    0    2    0    0   -1    0
    0    0    3    0    0   -1
eig(H)
ans =
        -1 + 1i
        -1 - 1i
        -1 + 3i
        -1 - 3i
        -1 + 2i
        -1 - 2i
```

house **Матрица Хаусхолдера**

Синтаксис:

[v, beta] = gallery('house', x)

Описание:

Функция [v, beta] = gallery('house', x) для вектора-столбца x длиной n возвращает вектор v и скаляр beta такие, что матрица H = eye(n, n) - beta*v*v' является матрицей Хаусхолдера.

Матрица Хаусхолдера удовлетворяет соотношению

H*x = -sign(x(1))*norm(x)*e1,

где e1 = [1 0 0 ...]'.

Заметим, что для комплексного числа z знак определяется соотношением sign(z) = exp(i*angle(z)), что совпадает со значением z./abs(z), если z не равно 0.

Если x = 0, то v = 0 и beta = 1.

invhess **Обратная к матрице в верхней форме Хессенберга**

Синтаксис:

I = gallery('invhess', x, y)

Описание:

Функция I = gallery('invhess', x, y), где x - вектор длиной n, а y - вектор длиной n-1, возвращает матрицу, нижняя треугольная часть которой совпадает с ones(n, 1)*x', а верхняя - с [1 y]*ones(1, n). Если x скаляр, равный k, то он интерпретируется как вектор 1:k. По умолчанию вектор y равен -x(1:n - 1).

Матрица невырожденна, если x(1) ~= 0 и x(i+1) ~= y(i) для всех i, и ее обратная является матрицей в верхней форме Хессенберга.

invol **Инволютивная матрица**

Синтаксис:

I = gallery('invol', n)

Описание:

Функция I = gallery('invol', n) возвращает матрицу порядка n, которая удовлетворяет свойству инволютивности I*I = eye(n)) и является плохо обусловленной.

Матрицы A = (eye(n) - I)/2 и A = (eye(n) + I)/2 удовлетворяют свойству идемпотентности A*A = A.

ipjfact **Факториальные матрицы Ганкеля**

Синтаксис:

[I, d] = gallery('ipjfact', n, k)

Описание:

Функция [l, d] = gallery('ipjfact', n, k) возвращает в зависимости от значения параметра k две формы факториальных матриц Ганкеля порядка n с элементами

l(i, j) = (i+j)! (k = 0, по умолчанию)

l(i, j) = 1/(i+j)! (k = 1).

В качестве второго выходного параметра возвращается определитель соответствующей факториальной матрицы Ганкеля.

Точные выражения для обратных факториальных матриц Ганкеля приведены в работе Gover M. J. C. "The explicit inverse of factorial Hankel matrices" (Dept. of Mathematics. University of Bradford, 1993).

jordblock **Блок Жордана**

Синтаксис:

J = gallery('jordbloc', n, lambda)

Описание:

Функция J = gallery('jordbloc', n, lambda) возвращает блок Жордана порядка n с собственным значением lambda. По умолчанию lambda равно 1.

kahan **Матрица Кахана**

Синтаксис:

K = gallery('kahan', [m n], theta)

K = gallery('kahan', n, theta)

K = gallery(..., pert)

Описание:

Функция K = gallery('kahan', [m n], theta) возвращает прямоугольную матрицу Кахана в верхней трапецеидальной форме. Диапазон возможных значений параметра 0 < theta < pi; по умолчанию theta равно 1.2.

Функция K = gallery('kahan', n, theta) возвращает матрицу Кахана порядка n в верхней треугольной форме.

Функция K = gallery(..., pert) позволяет оценить влияние ошибок округления на возможные перестановки столбцов. Для этого диагональные элементы матрицы Кахана подвергаются возмущению pert*eps*diag([n:-1:1]). По умолчанию pert равно 25, что гарантирует отсутствие перестановок при работе в IEEE-арифметике с матрицами, порядок которых не превышает 90.

Матрица Кахана обладает интересными свойствами относительно оценки числа обусловленности и ранга.

kms **Теплицева матрица в форме KMS**

Синтаксис:

K = gallery('kms', n, rho)

Описание:

Функция K = gallery('kms', n, rho) возвращает симметрическую теплицеву матрицу порядка n в форме KMS (Kac-Murdock-Szego) с элементами K(i, j) = = rho^(abs(i - j)) для действительных значений rho. Для комплексных rho элементы, расположенные ниже главной диагонали, являются комплексно сопряженными к элементам, находящимся выше диагонали. По умолчанию rho равно 0.5.

Теплицева матрица в форме KMS имеет следующие свойства:

- существует разложение K = LDL', где L = inv(triw(n, -rho, 1)') и D(i, i) = = (1 - abs(rho)^2)*eye(n), за исключением элемента D(1, 1), который равен 1;
- является положительно определенной, если 0 < abs(rho) < 1;
- обратная матрица является трехдиагональной.

krylov — **Матрица Крылова**

Синтаксис:

K = gallery('krylov', A, x, j)
K = gallery('krylov', n)

Описание:

Функция K = gallery('krylov', A, x, j) возвращает прямоугольную матрицу Крылова размера n×m вида [x, Ax, A^2x, ..., A^(m - 1)x], где A - произвольная матрица порядка n и x вектор длины n. По умолчанию x = ones(n, 1) и m = n.

Обращение в форме K = gallery('krylov', n) равносильно обращению gallery('krylov', randn(n)).

lauchli — **Матрица Лаучли**

Синтаксис:

L = gallery('lauchli', n, mu)

Описание:

Функция L = gallery('lauchli', n, mu) возвращает прямоугольную матрицу Лаучли (Lauchli) размера (n+1)×n вида [ones(1, n); mu*eye(n)]. Параметр mu по умолчанию равен sqrt(eps).

Матрица Лаучли дает хорошо известный пример проблемы, которая возникает в методе наименьших квадратов и других смежных областях и связанной с формированием произведения A'*A.

lehmer — **Матрица Лехмера**

Синтаксис:

L = gallery('lehmer', n)

Описание:

Функция L = gallery('lehmer', n) возвращает симметрическую положительно определенную матрицу Лехмера (Lehmer) порядка n с элементами A(i, j) = i/j для j ≥ i.

Матрица Лехмера обладает следующими свойствами:

- все элементы неотрицательные числа;
- обратная матрица трехдиагональная и известна точно;
- число обусловленности лежит в пределах n ≤ cond(L) ≤ 4*n*n.

lesp — **Матрица с действительными собственными значениями высокой чувствительности**

Синтаксис:

L = gallery('lesp', n)

Описание:

Функция L = gallery('lesp', n) возвращает матрицу порядка n, собственные значения которой равномерно расположены в интервале [-2*n - 3.5 - 4.5].

Эта матрица подобна симметрической трехдиагональной матрице с теми же элементами на главной диагонали и единицами на поддиагонали и наддиагонали; преобразование подобия реализуется диагональной матрицей D = diag(1!, 2!, ..., n!).

Чувствительность собственных значений возрастает экспоненциально по мере роста количества минимальных отрицательных значений.

lotkin — **Матрица Лоткина**

Синтаксис:

L = gallery('lotkin', n)

Описание:

Функция L = gallery('lotkin', n) возвращает матрицу Гильберта порядка n, у которой первая строка заменена единицами. Матрица Лоткина - это несимметрическая плохо обусловленная матрица с большим числом отрицательных собственных значений малой величины. Обратная матрица является целочисленной и точно известна.

Пример:

```
L =  gallery('lotkin', 4)
L =
    1      1      1      1
   1/2    1/3    1/4    1/5
   1/3    1/4    1/5    1/6
   1/4    1/5    1/6    1/7
```

```
inv(L)
ans =
    -4      120     -480      420
    30     -600     2700    -2520
   -60      900    -4320     4200
    35     -420     2100    -2100

eig(L)
ans =
  1115/591
 -379/1914
  -29/2361
  -15/104071
```

minij — **Симметрическая положительно определенная матрица**

Синтаксис:

M = gallery('minij', n)

Описание:

Функция M = gallery('minij', n) возвращает симметрическую положительно определенную матрицу порядка n с элементами M(i, j) = min(i, j).

Эта матрица обладает следующими свойствами:

- обратная матрица inv(M) является трехдиагональной;
- матрица Гивенса 2*M - ones(size(M)) имеет в качестве обратной трехдиагональную, а собственные значения матрицы Гивенса равны 0.5*sec((2*r-1)*pi/(4*n))^2, где r = 1:n;
- матрица (n+1)*ones(size(M)) - M имеет элементы max(i, j), а ее обратная является трехдиагональной.

moler — **Матрица Моулера**

Синтаксис:

M = gallery('moler', n, alpha)

Описание:

Функция M = gallery('moler', n, alpha) возвращает симметрическую положительно определенную матрицу порядка n, такую, что U'*U, где U = triw(n, alpha). По умолчанию alpha = -1 и M(i, j) = min(i, j) - 2,M(i, i) = i. При этом одно из собственных значений имеет очень малую величину.

Пример:

```
M = gallery('moler', 5)
M =
     1    -1    -1    -1    -1
    -1     2     0     0     0
    -1     0     3     1     1
    -1     0     1     4     2
    -1     0     1     2     5

eig(M)
ans =
    2.2785
    2.3965
    2.8289
 0.0086463
    7.4875
```

neumann **Матрица Неймана**

Синтаксис:

```
N = gallery('neumann', [m n])
N = gallery('neumann', n)
```

Описание:

Функция N = gallery('neumann', [m n]) возвращает сингулярную с диагональным преобладанием по строкам матрицу Неймана размера m×n, которая возникает в дискретной задаче Неймана с пятиточечным оператором на регулярной сетке.

Функция N = gallery('neumann', n) возвращает матрицу Неймана только для значений n, являющихся полным квадратом некоторого целого числа.

Матрица Неймана формируется в разреженном виде и имеет одномерное нуль-пространство, характеризуемое вектором ones(n, 1).

orthog **Ортогональные и близкие к ним матрицы**

Синтаксис:

```
O = gallery('orthog', n, k)
```

Описание:

Функция O = gallery('orthog', n, k) возвращает в зависимости от значения параметра k одну из матриц порядка n, указанную в таблице. При k > 0 формируется строго ортогональная матрица, а при k < 0 - ортогональная матрица с диагональным масштабированием. По умолчанию k равно 1.

k	*Описание матрицы*
1	O(i, j) = sqrt(2/(n+1)) * sin(i*j*pi/(n+1))
2	O(i, j) = 2/(sqrt(2*n+1)) * sin(2*i*j*pi/(2*n+1)) Симметрическая матрица
3	O(r, s) = exp(2*pi*i*(r-1)*(s-1)/n) / sqrt(n) Матрица Фурье, унитарная, O^4 - единичная матрица. Эта матрица совпадает с fft(eye(n))/sqrt(n)
4	Матрица Хелмерта в нижней форме Хессенберга, первая строка которой равна ones(1:n)/sqrt(n)
5	O(i, j) = sin(2*pi*(i-1)*(j-1)/n) + cos(2*pi*(i-1)*(j-1)/n) Симметрическая матрица, возникающая при работе с преобразованием Хартли
-1	O(i, j) = cos((i-1)*(j-1)*pi/(n-1)) Матрица Вандермонда для полиномов Чебышева, возникающая при отыскании экстремума полинома T(n -1)
-2	O(i, j) = cos((i-1)*(j-1/2)*pi/n)) Матрица Вандермонда для полиномов Чебышева, возникающая при отыскании нулей полинома T(n)

parter **Теплицева матрица с сингулярными значениями, близкими к π**

Синтаксис:

```
P = gallery('parter', n)
```

Описание:

Функция P = gallery('parter', n) возвращает матрицу P порядка n с элементами P(i, j) = 1/(i - j + 0.5). Матрица является матрицей Коши и теплицевой матрицей одновременно; часть ее максимальных сингулярных чисел расположена вблизи значения π.

Пример:

```
svd(gallery('parter',8) )
ans =
        3.14159265347603
        3.14159263512181
        3.14159131936263
        3.14153629454613
        3.14005961337616
        3.11402050998757
        2.83146652738175
        1.3490247654051
```

Сравните со значением pi = 3.14159265358979.

pie **Симметрическая матрица**

Синтаксис:

P = gallery('pei', n, alpha)

Описание:

Функция P = gallery('pei', n, alpha) возвращает симметрическую матрицу P = alpha*eye(n) + ones(n). По умолчанию alpha равно 1. Матрица ингулярна при значениях alpha, равных 0 и -n.

poisson **Матрица Пуассона**

Синтаксис:

P = gallery('poisson', n)

Описание:

Функция P = gallery('poisson', n) возвращает разреженную блочную трехдиагональную матрицу порядка n^2, которая возникает в связи с дискретизацией уравнения Пуассона на основе пятиточечной аппроксимации на сетке размера n×n.

prolate **Симметрическая плохо обусловленная теплицева матрица**

Синтаксис:

P = gallery('prolate', n, w)

Описание:

Функция P = gallery('prolate', n, w) возвращает симметрическую теплицеву матрицу порядка n со следующими свойствами:

- при значениях параметра 0 < w < 0.5 матрица P положительно определенная и плохо обусловленная;
- собственные значения различные, лежат в интервале (0, 1) и группируются вблизи 0 и 1;
- по умолчанию значение w равно 0.25.

randhess **Случайная ортогональная матрица в верхней форме Хессенберга**

Синтаксис:

R = gallery('randhess', n)
R = gallery('randhess', x)

Описание:

Функция R = gallery('randhess', n) возвращает действительную, случайную, ортогональную матрицу в верхней форме Хессенберга порядка n.

Функция R = gallery('randhess', x), где x - произвольный действительный вектор длины n > 1, возвращает детерминированную ортогональную матрицу.

Матрица R формируется как произведение n-1 вращения Гивенса.

rando **Случайная матрица из элементов -1, 0, 1**

Синтаксис:

R = gallery('rando', [m n], k)
R = gallery('rando', n, k)

Описание:

Функция R = gallery('rando', [m n], k) возвращает случайную матрицу размера m×n со следующими свойствами, зависящими от параметра k:

k	*Описание матрицы*
1	A(i, j) = 0 или 1 с равной вероятностью; используется по умолчанию
2	A(i, j) = -1 или 1 с равной вероятностью
3	A(i, j) = -1, 0 или 1 с равной вероятностью

Функция R = gallery('rando', n, k) возвращает случайную матрицу порядка n с теми же свойствами.

randsvd **Случайная матрица с предопределенными сингулярными числами**

Синтаксис:

R = gallery('randsvd', [m n], kappa, mode, kl, ku)
R = gallery('randsvd', n, kappa, mode, kl, ku)

Описание:

Функция R = gallery('randsvd', [m n], kappa, mode, kl, ku) возвращает ленточную случайную матрицу размера m×n с числом обусловленности cond(R) = kappa и распределением сингулярных чисел, зависящих от параметра mode:

mode	*Описание матрицы*
1	Одно большое сингулярное число
2	Одно малое сингулярное число
3	Сингулярные числа в геометрической прогрессии. Применяется по умолчанию
4	Сингулярные числа в арифметической прогрессии
5	Случайные сингулярные числа, логарифм которых распределен равномерно
< 0 (-1,...,-5)	Эти опции аналогичны abs(mode), но диагональные элементы матрицы сингулярных чисел, используемой в алгоритме, расположены в обратном порядке

По умолчанию kappa равно sqrt(1/eps).

Параметры kl и ku определяют количество используемых нижних и верхних поддиагоналей; если эти параметры опущены, матрица предполагается полной; если указано только kl, то по умолчанию ku = kl.

Функция R = gallery('randsvd', n, kappa, mode, kl, ku) возвращает ленточную случайную матрицу порядка n.

Особым является случай kappa < 0. При этом R - случайная, полная, симметрическая, положительно определенная матрица с числом обусловленности cond(A) = - kappa. В этом случае не сингулярные числа, а собственные значения имеют распределения, указанные в таблице. Параметры kl и ku не учитываются.

redheff **Матрица Редхеффера**

Синтаксис:

R = gallery('redheff', n)

Описание:

Функция R = gallery('redheff', n) возвращает матрицу Редхеффера порядка n, составленную из нулей и единиц с элементами R(i, j) = 1, если j = 1 или j делится на i, и R(i, j) = 0 в остальных случаях.

Матрица Редхеффера обладает следующими свойствами:

- она имеет (n - floor(log2(n))) - 1 собственных значений, равных 1;
- максимальное действительное собственное значение (спектральный радиус), равное приблизительно sqrt(n);
- остальные собственные значения достаточно малы;
- справедлива гипотеза Римана, если det(R) = O(n^(1/2+epsilon)) для любого epsilon > 0.

В работе [1, с. 673-683] авторы делают предположение, что все малые собственные значения находятся внутри круга единичного радиуса и что доказательство этого предположения вместе с утверждением, что собственное значение имеет тенденцию стремитья к нулю, когда n стремится к бесконечности, позволяет получить новое доказательство теоремы о простых числах.

Пример:

```
R =  gallery('redheff', 10)
R =
     1     1     1     1     1     1     1     1     1     1
     1     1     0     1     0     1     0     1     0     1
     1     0     1     0     0     1     0     0     1     0
     1     0     0     1     0     0     0     1     0     0
     1     0     0     0     1     0     0     0     0     1
     1     0     0     0     0     1     0     0     0     0
     1     0     0     0     0     0     1     0     0     0
     1     0     0     0     0     0     0     1     0     0
     1     0     0     0     0     0     0     0     1     0
     1     0     0     0     0     0     0     0     0     1
eig(R)
```

```
ans =
   4.3840
  -1.4224
   0.1888
   0.8496
   1.0000
   1.0000
   1.0000
   1.0000
   1.0000
   1.0000
see(R)
```

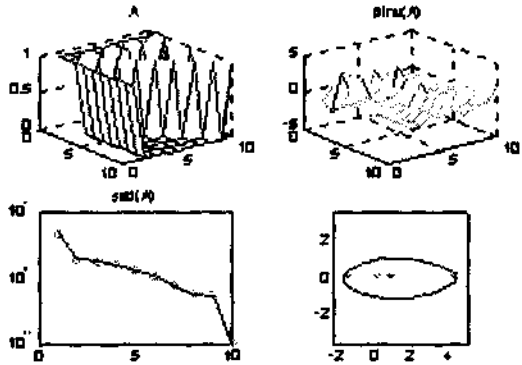


Рис. 7.5

Ссылки:

1. Barrett W. W., Jarvis T. J. Spectral Properties of a Matrix of Redheffer//Linear Algebra and Appl. 1992. Vol. 162.

riemann **Матрица Римана**

Синтаксис:

R = gallery('riemann', n)

Описание:

Функция R = gallery('riemann', n) возвращает матрицу порядка n, для которой выполняется гипотеза Римана в том и только в том случае, если det(R) = O(n! n^(-1/2+epsilon)) для любого epsilon > 0.

Матрица Римана определяется следующим образом:

R = A(2 : n+1, 2 : n+1), где A(i, j) = 1, если j делится на i, и A(i, j) = -1 в остальных случаях.

Матрица Римана обладает следующими свойствами:

- каждое собственное значение e(i) удовлетворяет условию abs(e(i)) ≤ m - 1/m, где m = n+1;

- все собственные значения e(i), за исключением самое большее m-sqrt(m), принадлежат интервалу i ≤ e(i) ≤ i+1;
- все целые числа в интервале (m/3, m/2] являются собственными значениями.

Пример:

```
R = gallery('riemann',20);
e = eig(R);
```

После соответствующей сортировки собственные значения можно расположить так, чтобы они отвечали вышеприведенным свойствам:

```
e =
 -0.1027
  1.3795 + 1.5150i
  1.3795 - 1.5150i
  3.0945
  4.8149
  5.3837
  6.8616
  8.0000
  9.0000
 10.0000
 11.1352
 12.6039
 13.3090
 14.4762
 15.4080
 16.7276
 17.4142
 18.7933
 19.5153
 20.8063
see(R)
```

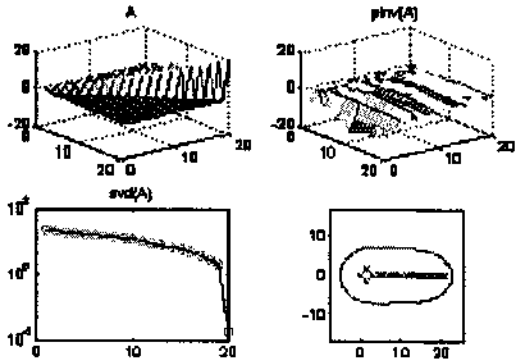


Рис. 7.6

ris **Матрица Риса**

Синтаксис:

```
R = gallery('ris', n)
```

Описание:

Функция R = gallery('ris', n) возвращает симметрическую ганкелеву матрицу порядка n с элементами A(i, j) = 0.5/(n - i- j + 1.5).

Собственные значения матрицы Риса группируются вблизи точек $\pi/2$ и $-\pi/2$. Матрица была предложена Рисом (F.N. Ris) и описана в работе [1].

Пример:

```
R =  gallery('ris', 20);
eig(R)
ans =
        -1.5707963267949
        -1.5707963267949
        -1.5707963267949
        -1.5707963267949
         1.5707963267949
         1.5707963267949
         1.5707963267949
        -1.57079632679455
         1.57079632679489
        -1.57079632648787
         1.57079632678376
         1.57079631958981
        -1.57079618238137
         1.57079385264652
        -1.57076016592562
         1.57034802437344
        -1.56614089140585
         1.53152701977956
        -1.32312990307175
        0.585252301226976
see(R, -1)
```

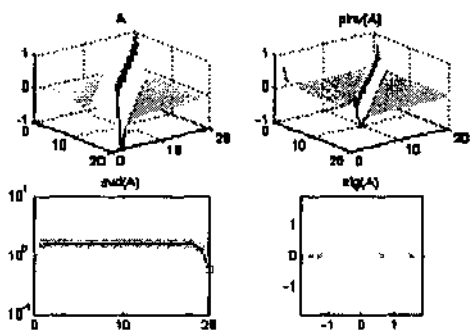


Рис. 7.7

Ссылки:

1. Nash J.C. Appendix 1//Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation. 2nd ed. Bristol: Adam Hilger, 1990.

smoke **Комплексная матрица с характерным псевдоспектром**

Синтаксис:

```
S = gallery('smoke', n)
S = gallery('smoke', n, 1)
```

Описание:

Функция S = gallery('smoke', n) возвращает матрицу порядка n с единицами на первой наддиагонали и в позиции (n, 1), а также комплексными корнями n-й степени из 1 вдоль главной диагонали. Собственные значения этой матрицы являются комплексными корнями n-й степени из 1, умноженными на 2^(1/n).

Функция S = gallery('smoke', n, 1) возвращает такую же матрицу, но только элемент в позиции (n, 1) равен 0. Собственные значения этой матрицы являются комплексными корнями n-й степени из 1.

Пример:

```
S = gallery('smoke', 10);
ps(S)
```

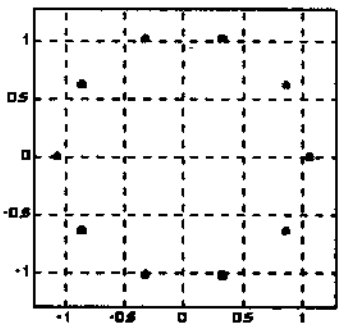


Рис. 7.8

toeppd **Симметрическая положительно полуопределенная теплицева матрица**

Синтаксис:

```
T= gallery('toeppd', n, m, w, theta)
```

Описание:

Функция T= gallery('toeppd', n, m, w, theta) возвращает симметрическую положительно полуопределенную теплицеву матрицу порядка n, сформированную как сумма m аналогичных матриц ранга 2 (а для некоторых theta ранга 1). По умолчанию m = n, w = rand(m, 1), theta = rand(m, 1).

Матрица формируется следующим образом:

T = w(1)*T(theta(1)) + ... + w(m)*T(theta(m)),

где (i, j) элемент матрицы T(theta(k)) равен cos(2*pi*theta(k)*(i - j)).

toeppen **Пятидиагональная разреженная теплицева матрица**

Синтаксис:

T= gallery('toeppen', n, a, b, c, d, e)

Описание:

Функция T = gallery('toeppen', n, a, b, c, d, e) возвращает пятидиагональную разреженную теплицеву матрицу порядка n с постоянными значениями на соответствующих диагоналях. Значения этих постоянных могут быть заданы значениями элементов T(3, 1) = a, T(2, 1) = b, T(1, 1) = c, T(1, 2) = d, T(1, 3) = e. По умолчанию (a, b, c, d, e) = (1, -10, 0, 10, 1), что соответствует матрице Рутисхаузера.

Эта матрица имеет собственные значения, расположенные на сегменте 2*cos(2*t) + 20*i*sin(t).

Пример:

```
T = gallery('toeppen', 10);
ps(T)
```

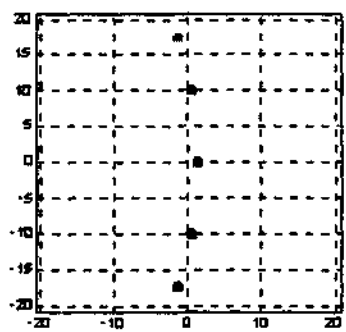


Рис. 7.9

tridiag **Трехдиагональная разреженная матрица**

Синтаксис:

T= gallery('tridiag', c, d, e)

T = gallery('tridiag', n, c, d, e)

Описание:

Функция T = gallery('tridiag', c, d, e) возвращает трехдиагональную разреженную матрицу с поддиагональю c, диагональю d и наддиагональю e. Векторы c и e должны иметь длину length(d) - 1.

Функция T = gallery('tridiag', n, c, d, e), где c, d, e - скаляры, возвращает трехдиагональную разреженную теплицеву матрицу порядка n с поддиагональными элементами c, диагональными - d и наддиагональными - e. Эта матрица имеет следующие собственные значения: d + 2*sqrt(c*e)*cos(k*pi/(n+1)), где k = 1 : n, приведенные в работе [1, с.155].

Оператор T = gallery('tridiag', n) равносилен оператору T = gallery('tridiag', n, -1, 2, -1), что определяет симметрическую положительно определенную M-матрицу.

Ссылки:

1. Todd J. Basic Numerical Mathematics. Vol. 2: Numerical Algebra. Basel; New York: Birkhauser: Academic Press, 1977.

triw **Верхняя треугольная матрица**

Синтаксис:

T= gallery('triw',[m n], alpha, k)
T = gallery('triw', n, alpha, k)

Описание:

Функция T= gallery('triw', [m n], alpha, k) возвращает прямоугольную верхнюю трапецеидальную матрицу с единицами на диагонали и значениями alpha на первых k наддиагоналях.

Функция T= gallery('triw', n, alpha, k) возвращает верхнюю треугольную матрицу порядка n. По умолчанию alpha = -1, k = n - 1. Эта матрица была предметом дискуссии в работах Кахана, Голуба и Уилкинсона [1-4].

В работе [3] показано, что число обусловленности cond(gallery('triw', n, 2)) = = cot(pi/(4*n))^2, а для больших значений abs(alpha) число обусловленности cond(gallery('triw', n, alpha)) приблизительно равно abs(alpha)^n*sin(pi/(4*n-2)).

Возмущение элемента T(n, 1) на величину -2^(2-n) делает матрицу сингулярной, так же как и возмущение всех элементов первой строки на величину -2^(1-n).

Ссылки:

1. Golub G. H., Wilkinson J. H. Ill-conditioned eigensystems and the computation of the Jordan canonical form// SIAM Review. 1976. Vol. 18(4). P. 578-619.
2. Kahan W. Numerical linear algebra// Canadian Math. Bulletin. 1966. Vol. 9. P. 757-801.
3. Ostrowski A. M. On the spectrum of a one-parametric family of matrices// J. Reine Angew. Math. 1954. Vol. 193 (3/4). P. 143-160.
4. Wilkinson J. H. Singular-value decomposition - basic aspects// Numerical Software - Needs and Availability/ D. A. H. Jacobs, ed. London: Academic Press, 1978. P. 109-135.

wathen **Матрица, связанная с методом конечных разностей**

Синтаксис:

W = gallery('wathen', nx, ny)
W = gallery('wathen', nx, ny, 1)

Описание:

Функция W = gallery('wathen', nx, ny) возвращает случайную разреженную матрицу порядка n, где n = 3*nx*ny + 2*nx + 2*ny + 1.

Это симметрическая положительная определенная матрица, соответствующая схеме двумерной восьмиточечной аппроксимации на сетке размера nx×ny [1].

Функция W = gallery('wathen', nx, ny, 1) возвращает диагонально масштабированную матрицу, такую, что 0.25 <= eig(W) <= 4.5.

Ссылки:

1. Wathen A. J. Realistic eigenvalue bounds for the Galerkin mass matrix// IMA J. Numer. Anal. 1987. Vol. 7. P. 449-457.

wilk **Матрицы Уилкинсона**

Синтаксис:

[W, b] = gallery('wilk', n)

Описание:

Функция [W, b] = gallery('wilk', n) возвращает различные матрицы или линейные системы в зависимости от значения параметра n, которые предлагались и обсуждались Уилкинсоном в связи с разработкой численных алгоритмов линейной алгебры.

Функция [W, b] = gallery('wilk', 3) формирует линейную систему с матрицей W в верхней треугольной форме, иллюстрирующую погрешности решения [1].

Функция [W, b] = gallery('wilk', 4) формирует линейную систему с матрицей W в нижней треугольной форме, которая является плохо обусловленной [2].

Функция W = gallery('wilk', 5) возвращает симметрическую положительно определенную матрицу порядка 5, сформированную из матрицы Гильберта, так что W = hilb(6)(1:5,2:6)*1.8144 [3].

Функция W = gallery('wilk', 21) возвращает трехдиагональную матрицу Уилкинсона порядка 21, связанную с решением проблемы собственных значений [3].

Ссылки:

1. Wilkinson J. H. Error analysis of direct methods of matrix inversion// J. Assoc. Comput. Mach. 1961. Vol. 8. P. 281-330.
2. Wilkinson J. H. Rounding Errors in Algebraic Processes, Notes on Applied Science No. 32. London: Her Majesty's Stationery Office, 1963.
3. Уилкинсон Дж. Х. Алгебраическая проблема собственных значений: Пер. с англ. М.: Наука, 1970. 564 с.

Сопутствующие функции: MAGIC, HILB, INVHILB, HADAMARD, WILKINSON.

Пакет программ Test Matrix Toolbox

Коллекция тестовых матриц, включенных в систему MATLAB 5, основана на работах Николаса Хайема (Nicholas J. Higham), выполненных на факультете математики университета Манчестера (Англия). Подробное описание этих матриц приведено автором в отчете [1].

Test Matrix Toolbox - это пакет тестовых матриц, который включает 58 тестовых матриц для задач линейной алгебры.

Пакет доступен для копирования с ftp-сервера университета по адресу:

ftp://ftp.ma.man.ac.uk/pub/higham/testmatrix.tar.Z,

а также с анонимного ftp-сервера фирмы MathWorks по адресу:

ftp://ftp.mathworks.com/pub/contrib/linalg/testmatrix.

Отчет размещен в этом же каталоге под именем testmatrix.ps.

Отчет доступен также и с сайта университета по адресам

ftp://ftp.ma.man.ac.uk/pub/narep

http://www.ma.man.ac.uk/MCCM/MCCM.html.

Теоретические основы пакета представлены в книге [2].

Помимо самих тестовых матриц в состав пакета включено 5 программ для визуализации матриц. Эти программы позволяют часто выявить такие свойства матрицы, которые непросто, а зачастую и невозможно усмотреть, глядя на числовые значения ее элементов. Часто эти программы дают самый простой способ получения приятных картинок.

Ниже приведено описание этих программ, поскольку в систему MATLAB они не включены, но мы настоятельно рекомендуем скопировать эти модули из пакета или подключить пакет Test Matrix Toolbox к установленной версии.

see — **Визуализировать матрицу и ее характеристики**

Синтаксис:

```
see(A)
see(A, 1)
see(A, -1)
```

Описание:

Команда see(A) позволяет отобразить в графическом окне, состоящем из четырех подокон, следующие графики:

mesh(A)	mesh(pinv(A))
semilogy(svd(A), '-o')	fv(A)

mesh(A) - поверхность, образованная значениями элементов матрицы A;
mesh(pinv(A)) - поверхность, образованная значениями элементов матрицы, обратной или псевдообратной A;

semilogy(svd(A)) - график в полулогарифмическом масштабе сингулярных чисел матрицы A;

fv(A) - область расположения собственных значений, удовлетворяющая отношениям Рэлея с указанием собственных значений.

Команда see(A, 1) отображает в третьем подокне аппроксимацию псевдоспектра вместо сингулярных чисел.

Команда see(A, -1) отображает в четвертом подокне только собственные значения.

Если матрица комплексная, то строится поверхность, соответствующая только действительной части матрицы.

Если матрица разреженная, то отображается только структура spy(A).

Пример:

Визуализировать матрицу chebspec(8))^3 и ее характеристики

```
A = gallery('chebspec', 8);
see(A^3, -1)
```

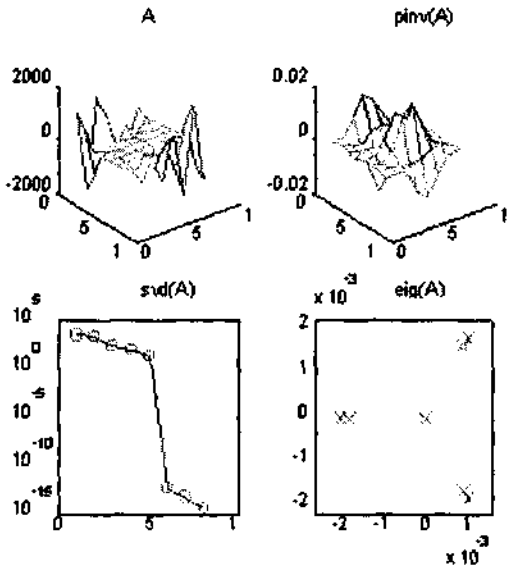


Рис. 7.10

fv **Построить область расположения собственных значений**

Синтаксис:

```
fv(A, nk, thmax)
[F, E] = fv(A, nk, thmax, 1)
```

Описание:

Команда fv(A, nk, thmax) вычисляет и строит область расположения собственных значений для nk ведущих подматриц произвольной комплексной матрицы A. Параметр thmax задает дискретность разбиения по угловой координате. По умолчанию nk = 1, thmax = 16. При подготовке публикаций рекомендуется устанавливать thmax = 32. Собственные значения отмечаются на графике знаком 'x'.

Функция [F, E] = fv(A, nk, thmax, 1) подавляет построение графика и возвращает область расположения в массиве F, а сами собственные значения - в массиве E.

Пример:

Построить область расположения собственных значений для матрицы grcar порядка 20.

```
A = gallery('grcar', 20);
fv(A, 1, 32)
```

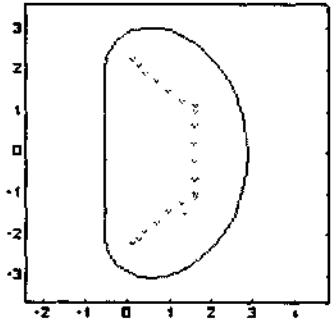


Рис. 7.11

gersh **Построение кругов Гершгорина**

Синтаксис:

```
gersh(A)
[G, E] = gersh(A, 1)
```

Описание:

Команда gersh(A) строит круги Гершгорина для произвольной квадратной комплексной матрицы в соответствии с теоремой Гершгорина, которая утверждает, что собственные значения матрицы A содержатся в некотором объединении круговых областей

$$D_i = \left\{ z \in \mathbf{C} : |z - a_{ii}| \le \sum_{\substack{j=1 \\ j \ne i}}^{n} |a_{ij}| \right\}.$$

Следствием теоремы Гершгорина является утверждение: если k кругов образуют связную область, которая изолирована от других кругов, то в этой области содержится точно k собственных значений.

Собственные значения отмечаются на графике знаком 'x'.

Функция [G, E] = gersh(A, 1) подавляет построение графика и возвращает данные графика в массиве G, а сами собственные значения - в массиве E.

Пример:

Построить круги Гершгорина для матрицы lesp порядка 20.

```
A = gallery('lesp', 20);
gersh(A)
```

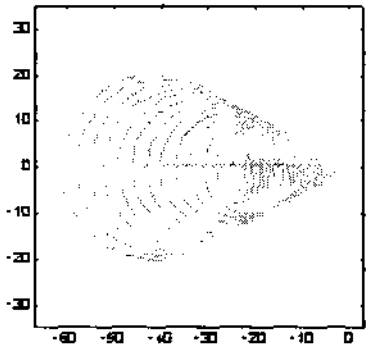


Рис. 7.12

Две оставшиеся графические функции ps и pscont связаны с построением ε-псевдоспектра матрицы $A \in C^{n \times n}$, который определяется как множество Λ_ε собственных значений возмущенных матриц A + E, для всех матриц E с $\|E\|_2 \le \varepsilon$.

ps **Точечный график псевдоспектра**

Синтаксис:

```
ps(A, m, tol, rl)
ps(A, m, tol, rl, marksize)
```

Описание:

Команда ps(A, m, tol, rl) строит приближенный график псевдоспектра для произвольной квадратной комплексной матрицы A, используя m случайных возмущений размера tol. По умолчанию m равно порядку матрицы, а tol равно 1e-3. Параметр rl определяет тип возмущения:

rl	*Тип возмущения*
0	По умолчанию; абсолютное комплексное возмущение с $\|E\|_2$ = tol
1	Абсолютное действительное возмущение с $\|E\|_2$ = tol
-1	Покомпонентное действительное возмущение размера tol

Собственные значения отмечаются на графике знаком 'x'.

Команда ps(A, m, tol, rl, marksize) использует указываемый пользователем размер маркера; если marksize < 0, то построение графика подавляется и возвращаются данные графика в виде выходного массива.

Команда ps(A, 0) строит график только собственных значений матрицы.

Пример:

Построить псевдоспектр пятидиагональной матрицы Теплица

```
A = gallery('toeppen', 32, 0, 1, 0, 0, 1/4);
ps(A)
```

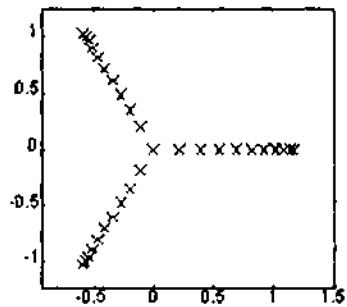


Рис. 7.13

pscont **Графики поверхностей и линий уровня псевдоспектра**

Синтаксис:

```
pscont(A, k, npts, ax, levels)
[X, Y, Z, npts] = pscont(A, ...)
```

Описание:

Команда pscont(A, k, npts, ax, levels) строит график функции log10(1/norm(R(z))), где R(z) = inv(z*I-A) - резольвента для квадратной матрицы A на сетке размером npts×npts [1, 2]. По умолчанию npts = round(min(max(5, sqrt(20^2*10^3/n^3)), 30)) и зависит от порядка матрицы. Параметр определяет тип графика:

k	*Тип графика*
0	По умолчанию; палитра цветов и линии уровня
1	Только линии уровня
2	Трехмерная поверхность и проекции линий уровня
3	Только трехмерная поверхность
4	Только линии уровня

Собственные значения отмечаются на графике знаком 'x'.

Вектор ax задает пределы изменения аргументов по осям x (ax(1), ax(2)) и y (ax(3), ax(4)). Если этот параметр опущен, границы вычисляются автоматически в зависимости от величин собственных значений матрицы.

Линии уровня задаются вектором levels, который по умолчанию равен -10 : -1, что соответствует значениям epsilon = 1e-10, ..., 1e-1.

Функция [X, Y, Z, npts] = pscont(A, ...) подавляет построение графика и возвращает данные в виде выходных массивов X, Y, Z и количества точек сетки npts.

Пример:

Построить поверхность и линии уровня псевдоспектра пятидиагональной матрицы Теплица

```
A = gallery('toeppen', 32, 0, 1, 0, 0, 1/4);
pscont(A, 2)
```

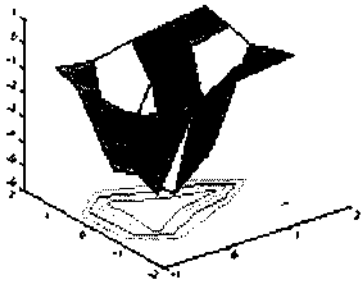


Рис. 7.14

Ссылки:

1. Kostin V. I. Linear algebra algorithms with guaranteed accuracy// Technical Report TR/PA/93/05. Toulouse: Cerfacs, 1993.
2. Trefethen L. N. Pseudospectra of matrices// eds. D.F. Griffiths and G.A. Watson, Numerical Analysis 1991. Proceedings of the 14th Dundee Conference. Vol. 260: Pitman Research Notes in Mathematics, Longman Scientific and Technical/ Essex, UK, 1992. P. 234-266.

В заключение следует отметить, что наряду с пакетом тестовых матриц Test Matrix Toolbox существуют и другие коллекции, среди которых следует отметить коллекции разреженных матриц [3], плотных матриц больших размеров для несимметрической проблемы собственных значений [4], а также прямоугольных матриц [5].

Ссылки:

1. Higham N. J. The Test Matrix Toolbox for MATLAB (version 3.0)//*Numerical Analysis Report*. Vol. 276. Manchester, 1995.
2. Higham N. J. Accuracy and Stability of Numerical Algorithms. SIAM. Philadelphia, PA, 1996. 690 pp.
3. Duff I. S., Grimes R. G., Lewis J. G. Users' guide for the Harwell-Boeing sparse matrix collection (release 1). *Report RAL-92-086*, Atlas Centre, Rutherford Appleton Labaratory, Didcot, Oxon, UK. 1992. P. 84.
4. Bai Z. *A collection of test matrices for large scale nonsymmetric eigenvalue problems* (version 1.0). Manuscript. 1994.
5. Zielke G. Report on test matrices for generalized inverses//*Computing*. 1986. Vol. 36. P. 105-162.

Характеристики матриц

DET **Определитель матрицы**

Синтаксис:

d = det(A)

Описание:

Функция d = det(A) вычисляет определитель квадратной матрицы; если матрица A целочисленная, то результатом является также целое число.

Алгоритм:

Определитель матрицы вычисляется на основе треугольного разложения методом исключения Гаусса:

[L, U] = lu(A);
s = det(L);
d = s * prod(diag(U)).

Пример:

Вычисление определителя и ранга матриц Гильберта различного порядка.

n	3	4	5	6	7	8	9	10	11
det	4.6296 e-004	1.6534 e-007	3.7493 e-012	5.3673 e-018	4.8358 e-025	2.7371 e-033	9.7203 e-043	2.1645 e-053	3.0274 e-065
rank	3	4	5	6	7	8	9	10	10

Сопутствующие функции: COND, INV, LU, RREF, /, \.

NORM **Нормы векторов и матриц**

Синтаксис:

n = norm(v, p)
n = norm(v)
n = norm(A, p)
n = norm(A, 'fro')
n = norm(A)

Описание:

Нормы векторов:

Функция n = norm(v, p) вычисляет p-норму вектора v, определяемую следующим образом:

$\| v \|_p$ = sum(abs(v).^p)^1/p.

Справедливы следующие соотношения:

norm(v) = norm(v, 2);
norm(v, inf) = max(abs(v));
norm(v, -inf) = min(abs(v)).

Нормы матриц:

Функция n = norm(A, p) вычисляет подчиненную p-норму матрицы A только для значений p, равных 1, 2, inf.

Справедливы следующие соотношения:
norm(A, 1) = max(sum(abs(A)));
norm(A, inf) = max(sum(abs(A')));
norm(A, 'fro') = sqrt(sum(diag(A' * A)));
norm(A) = norm(A, 2) = $\sigma_{max}(A)$.

Пример:

Нормы вектора:
v = -3 : 2;
norm(v, 1) = 9.0000
norm(v, 2) = 4.3589
norm(v, 3) = 3.5569
norm(v, inf) = 3
norm(v, -inf) = 0
norm(v) = 4.3589

Нормы матрицы:
A = toeplitz(1:5, 1.5:4.5);
norm(A, 1) = 15
norm(A, 2) = 12.1001
norm(A, inf) = 14
norm(A, 'fro') = 12.9422
norm(A) = 12.1001

Сопутствующие функции: COND, CONDEST, NORMEST.

NORMEST **Оценка 2-нормы матрицы**

Синтаксис:

nrm = normest(S)
nrm = normest(S, tol)
[nrm, count] = normest(...)

Описание:

Функция nrm = normest(S) вычисляет оценку 2-нормы матрицы S. Эта функция предназначена прежде всего для разреженных матриц, хотя она работает правильно и может быть полезна для больших полных матриц. Используйте эту функцию, когда вычисление нормы требует слишком большого времени, а для вас допустима приблизительная оценка.

Функция nrm = normest(S, tol) позволяет задать относительную погрешность вычислений взамен принятой по умолчанию величины 1.e–6.

Функция [nrm, count] = normest(...) возвращает оценку 2-нормы, а также количество использованных итераций.

Пример:

Матрица Уилкинсона порядка 101 - это трехдиагональная матрица достаточно высокого порядка, так что вычисление нормы norm(W) на основе svd-разложения занимает заметное время. Для ноутбука с частотой 50 МГц это составило 2.64 с и дало точное значение нормы 50.7462. Вычисление оценки для разреженной матрицы заняло всего 1.26 с и дало значение 50.7458.

```
W = wilkinson(101);
tic, norm(W), toc                    tic, normest((sparse(W))) , toc
ans =   50.7462                      ans =   50.7458
elapsed_time =  2.6400               elapsed_time =  1.2600
```

Сопутствующие функции: COND, CONDEST, NORM.

COND

Оценка числа обусловленности матрицы

Синтаксис:

```
k_1 = rcond(A)
```

Описание:

Функция k_1 = rcond(A) возвращает величину, обратную значению числа обусловленности матрицы A относительно 1-нормы, используя утилиту ZGECO пакета LINPACK [1]. Если матрица A хорошо обусловлена, то значение k_1 близко к единице; если матрица A плохо обусловлена, то значение k_1 близко к нулю.

Пример:

```
A = hilb(4);
1/rcond(A)          cond(A)          condest(A)
2.1523e+004         1.5514e+004      2.8375e+004
```

Сопутствующие функции: COND, CONDEST.

Ссылки:

1. Dongarra J. J., Bunch J. R., Moler C. B., Stewart G. W. LINPACK User's Guide. Philadelphia, 1979.

RANK

Ранг матрицы

Синтаксис:

```
r = rank(A)
r = rank(A, tol)
```

Описание:

Функция r = rank(A) возвращает ранг матрицы, который определяется как количество сингулярных чисел, превышающих порог max(size(A))*norm(A)*eps.

Функция r = rank(A, tol) возвращает ранг матрицы, который определяется как количество сингулярных чисел, превышающих заданный порог tol.

Алгоритм:

Существует несколько подходов к вычислению ранга матрицы. В системе MATLAB использован метод, основанный на вычислении сингулярных чисел матрицы A; он реализован в виде функции svd. Это наиболее надежный метод, хотя и требующий значительного времени на вычисление. Этот алгоритм идентичен используемому в пакете LINPACK [1].

Сам алгоритм вычисления ранга достаточно прост:

```
s = svd(A);
tol = max(size(A)) * s(1) * eps;
r = sum(s > tol).
```

Сопутствующие функции: SVD.

Ссылки:

1. Dongarra J. J., Bunch J. R., Moler C. B., Stewart G. W. LINPACK User's Guide. Philadelphia, 1979.

TRACE

След матрицы

Синтаксис:

```
t = trace(A)
```

Описание:

Функция t = trace(A) вычисляет след квадратной матрицы, равный сумме ее диагональных элементов.

Алгоритм:

Алгоритм вычисления следа матрицы на языке MATLAB - это однострочный М-файл

```
t = sum(diag(A)).
```

Пример:

Вычисление следа, определителя и ранга матриц Гильберта различного порядка.

n	3	4	5	6	7	8	9	10	11
trace	1.5333	1.6762	1.7873	1.8782	1.9551	2.0218	2.0806	2.1333	2.1809
det	4.6296 e-004	1.6534 e-007	3.7493 e-012	5.3673 e-018	4.8358 e-025	2.7371 e-033	9.7203 e-043	2.1645 e-053	3.0274 e-065
rank	3	4	5	6	7	8	9	10	10

Сопутствующие функции: DET, EIG.

NULL

Нуль-пространство (ядро) матрицы

Синтаксис:

```
Q = null(A)
Q = null(A, 'r')
```

Описание:

Функция Q = null(A) возвращает ортонормальный базис нуль-пространства матрицы A; если Q - не пустая матрица, то справедливы следующие соотношения:

```
Q' * Q = eye(size(A));
A * Q = 0.
```

Количество столбцов матрицы Q определяет размерность нуль-пространства или дефект матрицы A, что можно вычислить следующим образом:

defect = size(null(A), 2).

Функция Q = null(A, 'r') возвращает для целочисленной матрицы A базис нуль-пространства в рациональных числах, так называемый *рациональный базис*, вычисляемый с помощью функции rref.

На практике для численных расчетов предпочтительнее ортонормальный базис, а для методических целей - рациональный базис.

Пример:

Вычисление дефекта, следа, определителя и ранга матриц Гильберта различного порядка.

n	4	5	6	7	8	9	10	11	12
defect	0	0	0	0	0	0	0	0	1
trace	1.6762	1.7873	1.8782	1.9551	2.0218	2.0806	2.1333	2.1809	2.2244
det	1.6534 e-007	3.7493 e-012	5.3673 e-018	4.8358 e-025	2.7371 e-033	9.7203 e-043	2.1645 e-053	3.0274 e-065	2.7904 e-078
rank	4	5	6	7	8	9	10	10	11

Рассмотрим целочисленную матрицу

```
A =
     1   2   3
     1   2   3
     1   2   3
null(A) =                          null(A, 'r') =
   -0.1690  -0.9487                    -2   -3
    0.8452   0.0000                     1    0
   -0.5071   0.3162                     0    1
```

Сопутствующие функции: QR, ORTH, SUBSPACE.

ORTH **Ортонормальный базис матрицы**

Синтаксис:

Q = orth(A)

Описание:

Функция Q = orth(A) возвращает ортонормальный базис матрицы A; столбцы Q определяют то же пространство, которое определяют и столбцы A, но столбцы Q ортогональны, то есть

Q' * Q = eye(size(A)).

Количество столбцов матрицы Q определяет ранг матрицы A, что можно вычислить следующим образом:

rank = size(orth(A), 2).

Пример:

Вычисление ранга матриц Гильберта различного порядка:

n	4	5	6	7	8	9	10	11	12
rank	4	5	6	7	8	9	10	10	11

Сопутствующие функции: NULL, RANK, SVD.

SUBSPACE **Угол между двумя подпространствами**

Синтаксис:

theta = subspace(A, B)

Описание:

Функция theta = subspace(A, B) возвращает угол между двумя подпространствами, натянутыми на столбцы матриц A и B; если a и b - векторы единичной длины, то вычисляется угол между двумя векторами theta = acos(a' * b).

Если некоторая реализация физического эксперимента описывается массивом A, а другая реализация - массивом B, то функция subspace(A, B) определяет меру количества новой информации, полученной из второго эксперимента и не связанную со случайными ошибками.

Пример:

Рассмотрим два подпространства матрицы Адамара, столбцы которой, как известно, взаимно ортогональны.

```
H = hadamard(8);
A = H(:, 2:4);
B = H(:, 5:8);
size(A)
ans =    8    3
size(B)
ans =    8    4
```

Размеры этих подпространств различны, но для нахождения угла между ними не обязательно, чтобы их размеры точно совпадали. Геометрически ищется угол между двумя гиперплоскостями, погруженными в пространство более высокой размерности.

```
theta = subspace(A, B)
theta = 1.5708
```

Этот угол совпадает с $\pi/2$, в чем можно убедиться так:

```
theta – pi/2
ans = 0.
```

Сопутствующие функции: NULL, ORTH, QR.

RREF **Треугольная форма матрицы**

Синтаксис:

R = rref(A)	[R, jb] = rref(A)	rrefmovie(A)
R = rref(A, tol)	[R, jb] = rref(A, tol)	rrefmovie(A, tol)
		rrefmovie

Описание:

Функция R = rref(A) осуществляет приведение матрицы к треугольной форме на основе метода исключения Гаусса с частичным выбором ведущего элемента.

По умолчанию используется следующее значение порога для принятия решения о малости исключаемого элемента:

tol = max(size(A) * eps * norm(A, inf).

Функция R = rref(A, tol) осуществляет приведение матрицы к треугольной форме на основе метода исключения Гаусса с частичным выбором ведущего элемента для заданного значения порога tol.

Функции [R, jb] = rref(A) и [R, jb] = rref(A, tol) кроме треугольной формы возвращают также вектор jb, обладающий следующими свойствами:

- r = length(jb) может служить оценкой ранга матрицы A;
- при решении систем линейных уравнений Ax = b переменные x(jb) являются связанными переменными;
- столбцы A(:, jb) определяют базис матрицы A;
- R(1 : r, jb) - единичная.

Функции R = rrefmovie(A) и R = rrefmovie(A, tol) реализуют пошаговую процедуру приведения матрицы к треугольной форме с выводом на экран промежуточных матриц.

Функции R = rrefmovie демонстрируют пошаговую процедуру приведения некоторой матрицы размера 8 × 6 с рангом 4 к треугольной форме.

Примеры:

```
    A = magic(4), [R, jb]=rref(A)
A =                                R =
   16  2   3  13                      1   0   0   1
    5 11  10   8                      0   1   0   3
    9  7   6  12                      0   0   1  -3
    4 14  15   1                      0   0   0   0
    jb =    1        2        3
    r = length(jb)
    r =    3
A(:, jb) =                         R(:, jb) =
   16  2   3                         1   0   0
    5 11  10                         0   1   0
    9  7   6                         0   0   1
    4 14  15                         0   0   0
```

Сопутствующие функции: INV, LU, ORTH, RANK.

Решение линейных уравнений

\, /

Решатели систем линейных уравнений

Синтаксис:

X = B \ A

X = B / A

Описание:

Функция X = B \ A находит решение системы уравнений вида AX = B, где A - прямоугольная матрица размера m×n и B - матрица размера n×k.

Функция X = B / A находит решение системы уравнений вида XA = B, где A - прямоугольная матрица размера n×m и B - матрица размера m×k.

Алгоритм:

Решение систем линейных уравнений вида X = A \ B и X = B / A реализовано в MATLAB с помощью специального монитора, который использует разные алгоритмы решения в зависимости от структуры матрицы A.

- Если A - треугольная матрица с точностью до перестановки ее строк или столбцов, то решение таких систем уравнений может быть эффективно вычислено методом обратной подстановки. Проверка матрицы, является ли она верхней треугольной, осуществляется для полных матриц проверкой на нуль всех элементов, лежащих ниже диагонали; для разреженных матриц - определением структуры ее элементов. Большинство матриц нетреугольной структуры выявляются почти мгновенно, так что такая проверка требует очень малого времени.
- Если матрица A - симметрическая или эрмитова с положительными диагональными элементами, то применяется разложение Холецкого (функция chol). Если A - разреженная матрица, применяется алгоритм упорядочения по разреженности (функции symmmd и spparms). Если при этом матрица A положительно определенна, то алгоритм Холецкого позволяет эффективно найти решение. Матрицы, не являющиеся положительно определенными, выявляются почти мгновенно. Разложение Холецкого имеет вид:

  $A = L * L^T$,

  где L^T - верхняя треугольная матрица. После этого решение X можно получить решая последовательно две треугольные системы

  $X = L^T \backslash (L \backslash B)$.
- Если A - произвольная квадратная матрица, то треугольное разложение вычисляется методом исключения Гаусса с частичным выбором главного элемента (функция lu). Если A - разреженная матрица, применяется алгоритм упорядочения по разреженности столбцов (функции colmmd и spparms). В результате имеем следующее разложение:

  A = L * U,

где L - нижняя, а U - верхняя треугольные матрицы. После этого решение X можно получить решая последовательно две треугольные системы
X = U \ (L \ B).

- Если A - прямоугольная полная матрица, то применяется QR-разложение на основе преобразований Хаусхолдера следующего вида:
  A * P = Q * R,
  где P - матрица преобразований, Q - ортогональная и R - верхняя треугольная (функция qr) матрицы. Решение, соответствующее минимуму квадрата ошибки, находится согласно следующему соотношению:
  $X = P * (R \backslash (Q^T * B))$.
- Если A - прямоугольная разреженная матрица, то формируется вспомогательная расширенная матрица следующего вида:
  $S = [c*I \;\; A; \;\; A^T \;\; 0]$.
  Это реализуется с помощью функции spaugment. По умолчанию значение коэффициента масштабирования невязки c равно max(max(abs(A)))/1000 (функция spparms). Решение X в соответствии с методом наименьших квадратов и матрица невязок R = B - A * X вычисляются путем решения системы
  S * [R / c; X] = [B; 0]
  с использованием алгоритмов упорядочения по разреженности и исключения Гаусса с выбором главного элемента.
  Различные алгоритмы разложения матриц реализованы в системе MATLAB на основе ZGECO, ZGEFA и ZGESL для квадратных и процедур ZQRDC и ZQRSL для прямоугольных матриц из пакета LINPACK [1].

Диагностические сообщения:

При решении систем линейных уравнений:

если A - квадратная вырожденная матрица, выдается сообщение
Matrix is singular to working precision.
Для выбранной точности матрица вырожденна.

При поэлементном делении:

если массив-делитель имеет нулевые элементы, выдается сообщение
Divide by zero.
Деление на нуль.

На ЭВМ, где не реализован стандарт IEEE-арифметики, например на ЭВМ VAX, обе вышеприведенные операции будут выдавать сообщения об ошибке. На ЭВМ, где реализован стандарт IEEE-арифметики, например на PC, будут генерироваться только предупреждения. При этом при решении систем будут возвращаться матрицы, часть элементов которых будет иметь значение Inf; при поэлементном делении результатом могут быть как значения Inf, так и значение NaN.

Если результат обращения матрицы не является надежным, выдается сообщение

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

Предупреждение: Матрица близка к вырожденной или плохо масштабирована. Результаты могут быть неточными. Число обусловленности RCOND = xxx

При решении систем линейных уравнений:

если прямоугольная матрица A имеет неполный столбцовый ранг, выдается сообщение

Warning: Rank deficient, rank = xxx tol = xxx

Предупреждение: Неполный ранг, ранг rank = xxx точность tol = xxx

Сопутствующие функции: INV, QR, DET, LU, RCOND, ORTH, RREF.

Ссылки:

1. Dongarra J. J., Bunch J. R., Moler C. B., Stewart G. W. LINPACK User's Guide. Philadelphia, 1979.

INV **Обращение матрицы**

Синтаксис:

Y = inv(A)

Описание:

Функция Y = inv(A) вычисляет матрицу, обратную квадратной матрице A. В случаях, когда матрица A плохо масштабирована или близка к вырожденной, выдаются сообщения.

Функция реализована с использованием утилит ZGEDI и ZGEDA пакета LINPACK [1].

На практике вычисление явной обратной матрицы требуется не так часто. Как правило, говоря о задаче обращения, имеют в виду нахождение решений систем линейных уравнений. В рамках системы MATLAB для этих целей рекомендуется использовать решатели систем, то есть операторы вида x = A\b или x = b/ A, а не операцию x = inv(A)*b.

Диагностические сообщения:

В процессе выполнения функции inv на рабочих станциях и компьютерах с IEEE-арифметикой возможно появление следующего предупреждения:

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, появляется сообщение

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

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

Примеры:

Рассмотрим пример, демонстрирующий различие в определении решения систем линейных уравнений с помощью операций x = inv(A) * b и x = A \ b.

Сформируем матрицу Кахана из коллекции тестовых матриц со следующими характеристиками:

cond(A) = 1.6203e+010 norm(A) = 8.2970,

используя пакет Test Matrix Toolbox.

Затем сформируем результат в виде случайного вектора x = rand(100, 1);, а затем вычислим вектор b = A * x.

Используя компьютер PC/AT 486 с тактовой частотой 50 МГц, выполним следующие расчеты с фиксацией продолжительности вычислений:

```
A = gallery('kahan', 100, 1.35);
t0 = clock;                          t0 = clock;
y = inv(A) * b;                      z = A \ b;
t = etime(clock, t0)                 t = etime(clock, t0)
t =   1.0400                         t =   0.0600
err =  norm(y - x)                   err = norm(z - x)
err =  2.3224e-008                   err = 1.1668e-008
res = norm(A * y - b)                res = norm(A * z - b)
res =  2.5831e-008                   res = 1.0409e-014
```

Сравнивая оба подхода, можно убедиться, что расчеты с помощью решателя выполняются значительно быстрее и с большей точностью.

Сопутствующие функции: PINV, COND, CONDEST, LSCOV, NNLS, SLASH.

Ссылки:

1. Dongarra J. J., Bunch J. R., Moler C. B., Stewart G. W. LINPACK User's Guide. Philadelphia, 1979.

COND, CONDEST — Число обусловленности матрицы

Синтаксис:

```
k = cond(A)                          k = condest(A)
    k = cond(A, p)                   [k, v] = condest(A)
```

Описание:

Число обусловленности по отношению к операции обращения - это мера относительной погрешности, равная k = ||Δ(A-1)||/||ΔA||, где ||ΔA|| - норма матрицы погрешностей исходных данных. Оно характеризует точность операции обращения матрицы или решения системы линейных уравнений.

Функция k = cond(A) возвращает число обусловленности матрицы A по отношению к операции обращения, которое равно отношению максимального сингулярного числа к минимальному: $k = \sigma_{max}/\sigma_{min}$.

Функция k = cond(A, p) возвращает число обусловленности матрицы A относительно p-нормы k = norm(A, p) * norm(inv(A), p), где p = 1, 2, inf, 'fro'.

Функции cond(...) неприменимы для разреженных матриц.

Функция k = condest(A) вычисляет оценку нижней границы числа обусловленности по 1-норме.

Функция [k, v] = condest(A) вычисляет кроме оценки нижней границы числа обусловленности также и вектор, такой, что справедливо соотношение norm(A*v, 1) = norm(A, 1)*norm(v, 1)/k. Таким образом, вектор v может служить аппроксимацией нуль-вектора матрицы A, если k велико.

Функции condest(...) применимы для действительных, комплексных и разреженных матриц. В них реализован алгоритм Хейджера в модификации Хайема [2].

Пример:

Сравнительная оценка нижней границы числа обусловленности матрицы W=wilkinson(101) при применении функций cond, rcond, condest:

cond(W)	1/rcond(W)	condest(sparse(W))
ans = 199.9410	ans = 154.5382	ans = 276.2827

Наилучшая оценка нижней границы вычисляется функцией condest.

Сопутствующие функции: CONDEIG, NORM, NORMEST.

Ссылки:

1. Dongarra J. J., Bunch J. R., Moler C. B., Stewart G. W. LINPACK User's Guide. Philadelphia, 1979.
2. Higham N. J. Fortran Codes for Estimating the One-Norm of a Real orComplex Matrix, with Applications to Condition Estimation// ACM Trans. Math. Soft. Vol. 14. 1988. P. 381-396.

CHOL

Разложение Холецкого

Синтаксис:

R = chol(A)
[R, p] = chol(A)

Описание:

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; в противном случае появляется сообщение об ошибке.

Функция [R, p] = chol(A) никогда не генерирует сообщения об ошибке; если A - положительно определенная матрица, то p = 0 и матрица R совпадает с предшествующим случаем, в противном случае p > 0 и R - верхняя треугольная матрица порядка q = p -1, такая, что R' * R = A(1 : q, 1 : q).

Программа chol использует утилиту ZPOFA пакета LINPACK [1].

Примеры:

Рассмотрим матрицу Паскаля, составленную из биномиальных коэффициентов. Это положительно определенная матрица, а ее разложение Холецкого также использует часть биномиальных коэффициентов:

A = pascal(5)

A					R = chol(A)					eig(A)
1	1	1	1	1	1	1	1	1	1	0.0108
1	2	3	4	5	0	1	2	3	4	0.1812
1	3	6	10	15	0	0	1	3	6	1.0000
1	4	10	20	35	0	0	0	1	4	5.5175
1	5	15	35	**70**	0	0	0	0	1	92.2904

Разрушим положительную определенность этой матрицы, вычтя из ее последнего элемента единицу:

A					[R, p] = chol(A)				eig(A)
1	1	1	1	1	1	1	1	1	0.0000
1	2	3	4	5	0	1	2	3	0.1500
1	3	6	10	15	0	0	1	3	0.9522
1	4	10	20	35	0	0	0	1	5.3531
1	5	15	35	**69**	p = 5				91.5446

```
R' * R =
     1    1    1    1
     1    2    3    4
     1    3    6   10
     1    4   10   20
```

Сопутствующие функции: QR, LU.

Ссылки:

1. Dongarra J. J., Bunch J. R., Moler C. B., Stewart G. W. LINPACK User's Guide. Philadelphia, 1979.

CHOLUPDATE — Разложение Холецкого модифицированной матрицы

Синтаксис:

```
    R1 = cholupdate(R, x)
    R1 = cholupdate(R, x, '+')
    R1 = cholupdate(R, x, '–')
[R1, p] = cholupdate(R, x, '–')
```

Описание:

Если R разложение Холецкого матрицы A, то cholupdate(R, x) определяет разложение Холецкого для матрицы $A + x*x^T$, которая является суммой положительно определенной матрицы A и матрицы ранга 1.

Функция R1 = cholupdate(R, x) вычисляет разложение Холецкого матрицы A, модифицированной матрицей ранга 1, когда известно разложение Холецкого R матрицы A.

Функция R1 = cholupdate(R, x, '+') равносильна функции R1 = cholupdate(R, x).

Функция R1 = cholupdate(R, x, '-') возвращает разложение Холецкого для матрицы $A - x*x^T$. Сообщение об ошибке формируется в тех случаях, когда R не является разложением Холецкого матрицы A или если модифицированная матрица не является положительно определенной и не имеет разложения Холецкого.

Функция [R1, p] = cholupdate(R, x, '-') не возвращает сообщения. Если выходной параметр p равен 0, то R1 - разложение Холецкого для матрицы $A - x*x^T$; если p равно 1, то модифицированная матрица не является положительно определенной; если p равно 2, то модифицированная матрица не имеет разложения Холецкого.

Функции cholupdate(...) могут применяться только для полных матриц.

Программа chol использует утилиты ZCHUD и ZCHDD пакета LINPACK [1].

Примеры:

Рассмотрим матрицу Паскаля, составленную из биномиальных коэффициентов.

```
A = pascal(4)
A =
   1  1   1   1
   1  2   3   4
   1  3   6  10
   1  4  10  20
R = chol(A)
R =
1 1 1 1
0 1 2 3
0 0 1 3
0 0 0 1
x = [0 0 0 1]';
```

Добавим к матрице A матрицу ранга 1:

```
A + x*x'
ans =
1  1   1   1
1  2   3   4
1  3   6  10
1  4  10  21
R1 = cholupdate(R, x)
R1 =
1.0000 1.0000 1.0000 1.0000
   0   1.0000 2.0000 3.0000
   0      0   1.0000 3.0000
   0      0      0   1.4142
```

Попытаемся вычислить разложение Холецкого для модифицированной матрицы, которая равносильна вычитанию из последнего элемента 1:

```
R1 = cholupdate(R, x, '–')
??? Error using ==> cholupdate
Downdated matrix must be positive definite.
```

Модифицированная матрица должна быть положительно определенной.

Вычислим разложение Холецкого для модифицированной матрицы, которая равносильна вычитанию из последнего элемента 1/2:

```
x = [0 0 0 1/sqrt(2)]';
R1 = cholupdate(R, x, '–')
R1 =
1.0000 1.0000  1.0000  1.0000
   0   1.0000  2.0000  3.0000
   0      0    1.0000  3.0000
   0      0       0    0.7071
```

Сопутствующие функции: CHOL, QRUPDATE.

Ссылки:

1. Dongarra J. J., Bunch J. R., Moler C. B., Stewart G. W. LINPACK User's Guide. Philadelphia, 1979.

LU — LU-разложение

Синтаксис:

```
[L, U] = lu(A)
[L, U, P] = lu(A)
lu(A)
```

Описание:

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, так что A = L * U.

Функция [L, U, P] = lu(A) находит разложение для произвольной квадратной матрицы A в виде трех составляющих - нижней треугольной матрицы L, верхней треугольной матрицы U и матрицы перестановок P, так что P * A = L * U.

Функция lu(A) возвращает выход, формируемый утилитой ZGEFA пакета LINPACK.

Обращение к функции lu в форме lu(A, thresh) позволяет управлять выбором главного элемента при работе с разреженными матрицами, где параметр thresh выбирается из диапазона [0, 1] и позволяет установить порог выбора. Выбор главного элемента реализуется, когда диагональный элемент в столбце не превышает произведения величины внедиагонального элемента и параметра thresh. Значение thresh = 0 означает выбор в качестве ведущего диагонального элемента; значение thresh = 1 используется по умолчанию.

LU-разложение используется при вычислении определителей, нахождении обратных матриц и в решателях систем линейных уравнений.

Программа lu использует утилиты ZGEDI и ZGEFA пакета LINPACK [1].

Примеры:

Рассмотрим возмущенную отрицательно определенную матрицу Паскаля 3-го порядка следующего вида:

```
A =                                  eig(A) =
        1    1    1                    0.7024
        1    2    3                   -0.2185
        1    3    4                    6.5160
[L, U] = lu(A)
L =                        U =                          L * U =
  1.0000   0        0        1.0000   1.0000   1.0000     1    1    1
  1.0000   0.5000   1.0000   0        2.0000   3.0000     1    2    3
  1.0000   1.0000   0        0        0        0.5000     1    3    4
det(A) = det(L) * det(U) = -1;
[L, U, P] = lu(B)
L =                        U =                          P =          L * U =
  1.0000   0        0        1.0000   1.0000   1.0000     1  0  0     1  1  1
  1.0000   1.0000   0        0        2.0000   3.0000     0  0  1     1  3  4
  1.0000   0.5000   1.0000   0        0        0.5000     0  1  0     1  2  3
det(A) = det(L) * det(U)/det(P) = -1.
```

Сопутствующие функции: LUINC, QR, RREF.

Ссылки:

1. Dongarra J. J., Bunch J. R., Moler C. B., Stewart G. W. LINPACK User's Guide. Philadelphia, 1979.

QR, QRDELETE, QRINSERT

QR-разложение

Синтаксис:

Для полных матриц:	*Для разреженных матриц:*	
[Q, R] = qr(A)	R = qr(A)	[Q, R] = qrdelete(Q, R, j)
[Q, R, P] = qr(A)	[Q, R] = qr(A)	[Q, R] = qrinsert(Q, R, j, x)
[Q, R] = qr(A, 0)	[C, R] = qr(A, B),	
[Q, R, E] = qr(A, 0)	R = qr(A, 0)	
Y = qr(A)	[C, R] = qr(A, B, 0)	

Описание:

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

Функция [Q, R, P] = qr(A) находит разложение для прямоугольной матрицы A в виде трех составляющих - унитарной матрицы Q, верхней треугольной матрицы R с убывающими по модулю диагональными элементами и матрицы перестановок P, так что A * P = Q * R.

Функция [Q, R] = qr(A, 0) находит так называемое *экономичное* QR-разложение для матрицы A размера m×n при m > n, при котором вычисляется только n столбцов матрицы Q.

Функция [Q, R, e] = QR(A, 0) также формирует экономичное разложение и вектор возмущения e такой, что Q*R = [A e] и элементы abs(diag(R)) расположены в убывающем порядке.

Функция Y = qr(A) возвращает матрицу Y, являющуюся выходом утилиты ZQRDC пакета LINPACK [1], которая связана с матрицей R соотношением R = triu(Y).

Все перечисленные выше функции предназначены для работы с полными матрицами.

Для разреженных матриц QR-разложение может формировать только матрицу R, если реализуются следующие обращения:

функция R = qr(A) возвращает только матрицу R, причем R = chol(A'*A);

функция [Q, R] = qr(A) возвращает только матрицы Q и R, причем матрица Q является почти полной;

функция [C, R] = qr(A, B), где B имеет столько же строк, как и A, возвращает матрицу C = Q'*B;

функции R = qr(A, 0) и [C, R] = qr(A, B, 0) реализуют экономичное QR-разложение.

Версии QR-разложения для разреженных матриц не выполняют перестановок столбцов.

Решение системы линейных уравнений A*x = b методом наименьших квадратов может быть получено без вычисления матрицы Q с помощью такой последовательности операторов:

```
x = R\(R'\(A'*b))
r = b - A*x
e = R\(R'\(A'*r))
x = x + e;
```

Функция [Q, R] = qrdelete(Q, R, j) позволяет пересчитать известное QR-разложение матрицы A для случая, когда в матрице A удален j-й столбец A(:, j).

Функция [Q, R] = qrinsert(Q, R, j, x) позволяет пересчитать известное QR-разложение матрицы A для случая, когда в матрице A перед j-м столбцом A(:, j) вставлен дополнительный столбец x. Если указать j = n+1, где n - число столбцов матрицы A, то дополнительный столбец x будет n+1-м столбцом матрицы.

Примеры:

Рассмотрим операции, связанные с QR-разложением следующей прямоугольной матрицы A:

```
[Q, R] = qr(A)
```

```
A =                 Q =                                          R =
   1   2   3          -0.0776  -0.8331   0.5444   0.0605          -12.8841  -14.5916  -16.2992
   4   5   6          -0.3105  -0.4512  -0.7709   0.3251            0        -1.0413   -2.0826
   7   8   9          -0.5433  -0.0694  -0.0913  -0.8317            0         0         0.0000
  10  11  12          -0.7762   0.3124   0.3178   0.4461            0         0         0
```

Удалим 2-й столбец из матрицы A:

```
[Q1, R1] = qrdelete(Q, R, 2)
```

```
Q1 =                                        R1 =
  -0.0776  0.8331  -0.5444  0.0605          -12.8841-16.2992
  -0.3105  0.4512   0.7709  0.3251            0       2.0826
  -0.5433  0.0694   0.0913 -0.8317            0       0
  -0.7762 -0.3124  -0.3178  0.4461            0       0
```

Вставим 2-й столбец из матрицы A на место 3-го:

```
[Q2, R2] = qrinsert(Q1, R1, 3, A(:,2))
Q2 =                                  R2 =
-0.0776  0.8331  -0.5458  -0.0456     -12.8841-16.2992-14.5916    1.0000  3.0000  2.0000
-0.3105  0.4512   0.6938   0.4676      0        2.0826  1.0413    4.0000  6.0000  5.0000
-0.5433  0.0694   0.2499  -0.7985      0        0       0.0000    7.0000  9.0000  8.0000
-0.7762 -0.3124  -0.3979   0.3764      0        0       0        10.0000 12.0000 11.0000
```

Сопутствующие функции: LU, NULL, ORTH, QRDELETE, QRINSERT, QRUPDATE.

Ссылки:

1. Dongarra J. J., Bunch J. R., Moler C. B., Stewart G. W. LINPACK User's Guide. Philadelphia, 1979.

QRUPDATE — QR-разложение модифицированной матрицы

Синтаксис:

```
[Q1, R1] = qrupdate(Q, R, u, v)
```

Описание:

Функция [Q1, R1] = qrupdate(Q, R, u, v), где [Q, R] = qr(A) - QR-разложение исходной матрицы A, возвращает QR-разложение модифицированной матрицы A + u*v', где u и v векторы-столбцы соответствующей длины. Эта функция может применяться только для полных матриц.

Пример:

```
mu = sqrt(eps)
mu = 1.4901e-08
A = [ones(1, 4); mu*eye(4)];
```

Эта матрица позволяет увидеть проблемы, которые возникают в связи с формированием произведения A'*A. Чтобы избежать этих трудностей, воспользуемся QR-разложением матрицы A.

```
[Q, R] = qr(A);
R =
     -1          -1          -1          -1
      0  2.1073e-008  1.0537e-008  1.0537e-008
      0           0   1.825e-008   6.0834e-009
      0           0           0   1.7206e-008
      0           0           0             0
```

В данном случае верхняя треугольная матрица R, исключая первую строку, содержит элементы порядка O(mu).

Введем векторы

```
u = [-1 0 0 0 0]'; v = ones(4,1);
```

Теперь вычислим QR-разложение матрицы A, модифицированной матрицей ранга 1, прямым методом и сравним с результатом процедуры qrupdate:

```
[QT, RT] = qr(A + u*v')
QT =
   0   0   0   0   1
  -1   0   0   0   0
   0  -1   0   0   0
   0   0  -1   0   0
   0   0   0  -1   0
RT =
 -1.4901e-008          0          0          0
            0  -1.4901e-008          0          0
            0          0  -1.4901e-008          0
            0          0          0  -1.4901e-008
            0          0          0          0

[Q1, R1] = qrupdate(Q, R, u, v)
Q1 =
 -2.2352e-008  -1.4901e-008  -1.4901e-008  -1.4901e-008           1
            1  -3.3307e-016  -3.3307e-016  -3.0531e-016  2.2352e-008
 -1.6653e-016           1  -2.2204e-016  -1.6653e-016  1.4901e-008
 -2.7477e-016  -2.7477e-016           1  -1.6653e-016  1.4901e-008
            0           0           0           1  1.4901e-008
R1 =
  1.4901e-008  1.6544e-024  3.3087e-024  3.3087e-024
            0  1.4901e-008  -3.6734e-040  -3.6734e-040
            0           0  1.4901e-008  -1.6544e-024
            0           0           0  1.4901e-008
            0           0           0            0
```

Следует отметить, что оба разложения корректны, но отличаются одно от другого.

Сопутствующие функции: CHOLUPDATE, QR.

Ссылки:

1. Golub, Gene H., Charles Van Loan, Matrix Computations. 3rd ed. Baltimore: Johns Hopkins University Press, 1996.

PINV **Псевдообращение матрицы по Муру-Пенроузу**

Синтаксис:

P = pinv(A)
P = pinv(A, tol)

Описание:

Функция P = pinv(A) вычисляет матрицу, псевдообратную матрице A, которая имеет такие же размеры, как и матрица A', и удовлетворяет следующим условиям [1]:

A * P * A = A;
P * A * P = P.

Вычисление матрицы P основано на использовании функции svd(A) и приравнивании к нулю всех сингулярных чисел, меньших величины tol, которая по умолчанию принимается равной tol = max(size(A)) * norm(A) * eps.

Функция P = pinv(A, tol) позволяет пользователю самому назначить порог tol.

Если матрица A квадратная и невырожденная, то вычисление обратной на основе псевдообращения является слишком расточительной процедурой. Если A - квадратная и вырожденная или прямоугольная матрица, то обратной матрицы не существует; в этих случаях псевдообратная матрица обладает некоторыми, но не всеми свойствами обратной.

Если A имеет строк больше, чем столбцов, и не является матрицей полного ранга, то возникает переопределенная задача наименьших квадратов

$$\min_{x} \text{norm}(A * x - b).$$

Вектор x минимизирует указанную норму тогда и только тогда, когда x имеет вид

x = pinv(A) * b + (I - pinv(A) * A) * z

для некоторого z.

Выберем два из бесконечного множества решений:

x = pinv(A) * b;
y = A \ b.

Эти решения характеризуются следующими свойствами: решение x имеет норму norm(x), которая минимальна в сравнении с нормой любого другого решения; решение y имеет минимальное количество ненулевых компонентов.

Пример:

Рассмотрим прямоугольную матрицу, которая генерируется следующим образом:

A = magic(8); A = A(:, 1:6).

Эта матрица размером 8×6 имеет ранг, равный 3.

Сформируем вектор b = 260 * ones(8, 1).

Тогда получим следующие решения:

x = pinv(A) * b	y = A \ b
	Warning: Rank deficient, rank = 3 tol = 1.8829e-013
	Предупреждение: Ранг не полный, rank = 3 tol = 1.8829e-013
x =	y =
1.1538	3.0000
1.4615	4.0000
1.3846	0
1.3846	0
1.4615	1.0000
1.1538	0
norm(x) = 3.2817	norm(y) = 5.0990

Вектор z, удовлетворяющий условию y = x + (I - pinv(A) * A) * z, равен

z' = [5.7517 9.6751 4.6404 5.8559 7.8906 1.5362]

Сопутствующие функции: RANK, INV, SVD, QR.

Ссылки:

1. Алберт А. Регрессия, псевдоинверсия и рекуррентное оценивание: Пер. с англ. М.: Наука, 1977. 224 с.

NNLS **Метод наименьших квадратов с ограничениями**

Синтаксис:

x = nnls(A, b) [x, w] = nnls(A, b)

x = nnls(A, b, tol) [x, w] = nnls(A, b, tol)

Описание:

Функция x = nnls(A, b) находит неотрицательные решения $x_j \geq 0$, j = 1,..., n для системы уравнений вида Ax = b по методу наименьших квадратов. Для отбора таких решений по умолчанию используется значение порога tol = max(size(A)) * norm(A,1) * eps.

Функция x = nnls(A, b, tol) позволяет пользователю самому установить значение порога tol.

Функции [x, w] = nnls(A, b) и [x, w] = nnls(A, b, tol) позволяют в дополнение к решению x возвратить также вектор двойственных переменных w. Векторы x и w связаны между собой следующими соотношениями:

$w_i < 0$, $(i \mid x_i = 0)$;

$w_i = 0$, $(i \mid x_i > 0)$.

Пример:

Сравним решения задачи наименьших квадратов с ограничениями и без них. Пусть задана следующая система уравнений, описываемая парой {A, b}:

```
A =                                  b =
        0.0372    0.2869                 0.8587
        0.6861    0.7071                 0.1781
        0.6233    0.6245                 0.0747
        0.6344    0.6170                 0.8405
```

Вычислим решения без ограничений и с ограничениями на переменные, а также нормы невязок:

```
[   A\b          nnls(A, b)  ]     [norm(A * (A \ b) - b)     norm(A * nnls(A, b) - b)]
    -2.5627      0
     3.1108      0.6929                0.6674                     0.9118
```

Как следует из решения, невязка для решения без ограничений меньше, но при этом один из компонентов вектора x отрицателен.

Найдем вектор двойственных переменных w:

```
[x, w] = nnls(A, b)
x =                                  w =
        0                                   -0.1506
        0.6929                               0.0000
```

Сопутствующие функции: LSCOV, \.

Ссылки:

1. Lawson C. L., Hanson R. J. Solving Least Squares Problems. Prentice-Hall, 1974.

LSCOV **Метод наименьших квадратов в присутствии шумов**

Синтаксис:

```
x = lscov(A, b, V)
[x, dx] = lscov(A, b, V)
```

Описание:

Функция x = lscov(A, b, V) возвращает решение x для следующей системы уравнений: Ax = b + v, где вектор шумов v имеет матрицу ковариаций V. Решение минимизирует по методу наименьших квадратов следующую квадратичную форму:

(Ax - b)' * inv(V) * (Ax - b).

Решение задачи имеет вид [1, 2]:

x = inv(A' * inv(V) * A) * A' * inv(V) * b.

Реально алгоритм построен так, что обращения матрицы V не требуется.

Функция [x, dx] = lscov(A, b, V) возвращает стандартные погрешности решения x, вычисляемые по формулам

mse = B'*(inv(V)-inv(V)*A*inv(A'*inv(V)*A)*A'*inv(V))*B./(m-n),

dx = sqrt(diag(inv(A'*inv(V)*A)*mse))

в векторе dx.

Сопутствующие функции: QR, NNLS, \.

Ссылки:

1. Strang G. Introduction to Applied Mathematics. Wellesley-Cambridge, 1986.
2. Алберт А. Регрессия, псевдоинверсия и рекуррентное оценивание: Пер. с англ. М.: Наука, 1977. 224 с.

Вычисление собственных значений и сингулярных чисел

EIG, CDF2RDF **Собственные значения и собственные векторы матрицы**

Синтаксис:

d = eig(A)

[R, D] = eig(A)

[R, D] = eig(A, 'nobalance')

[R, D] = cdf2rdf(R, D)

d = eig(A, B)

[V, D] = eig(A, B)

Описание:

Проблема собственных значений состоит в нахождении нетривиальных решений системы уравнений, которая может быть интерпретирована как алгебраический эквивалент системы обыкновенных дифференциальных уравнений в явной форме Коши:

$Ar = \lambda r$,

где A - квадратная матрица порядка n;

r - вектор-столбец размера 1×n, называемый *собственным вектором*;

λ - скаляр, называемый *собственным значением.*

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

Функция [R, D] = eig(A) вычисляет диагональную матрицу D собственных значений и матрицу R правых собственных векторов, удовлетворяющих соотношению A * R = R * D. Эти векторы нормированы так, что норма каждого из них равна единице.

Левые собственные векторы могут быть найдены следующим образом:

[L, D] = eig(A');

Матрицы собственных значений D для A и A' содержат одни и те же собственные значения, хотя порядок их следования может быть различен. Матрица левых собственных векторов удовлетворяет соотношению A' * L = L * D. Для согласования независимо найденных систем правых и левых собственных векторов систему левых векторов необходимо нормировать так, чтобы соблюдалось условие L * R = eye(n,n).

Функция [R, D] = cdf2rdf(R, D) преобразовывает комплексные выходы функции eig в действительные, при этом комплексные собственные значения преобразовываются в блоки размера 2×2, а комплексная матрица правых собственных векторов R преобразовывается в действительную, столбцы которой, соответствующие действительным собственным значениям, сохраняются, а соответствующие комплексным - расщепляются на два: [Re(ri) Im(ri)].

Пример 1:

Рассмотрим матрицу порядка 3 с одним действительным и парой комплексно сопряженных собственных значений и выполним вычисления с использованием комплексных матриц).

Применяя функцию cdf2rdf, эти же вычисления можно реализовать, используя только действительные матрицы, что позволяет более экономно расходовать память).

Функция [R, D] = eig(A, 'nobalance') вычисляет собственные значения и собственные векторы без предварительного масштабирования матрицы. Обычно масштабирование улучшает обусловленность матрицы, гарантируя большую точность вычислений. Однако когда матрица содержит очень малые по величине элементы, которые находятся в пределах ошибок округления, масштабирование может сделать их сравнимыми с другими элементами матрицы, что в состоянии привести к неправильным результатам.

Пример 2:

Рассмотрим матрицу порядка 4, которая содержит элементы, сравнимые с ошибками округления.

$$B = \begin{bmatrix} 3 & -2 & -0.9 & 2*eps \\ -2 & 4 & -1 & -eps \\ -eps/4 & eps/2 & -1 & 0 \\ -0.5 & -0.5 & 0.1 & 1 \end{bmatrix};$$

```
[RB, DB]  = eig(B);
DB =
  5.5616  0       0       0
  0       1.4384  0       0
  0       0       1.0000  0
  0       0       0       -1.0000
norm(B * RB - RB * DB) = 1.24392
```

```
RN, DN] = eig(B, 'nobalance');
DN =
  5.5616  0       0       0
  0       1.4384  0       0
  0       0       1.0000  0
  0       0       0       -1.0000
norm(B*RN- RN*DN) = 0.9957e-015
```

Как следует из этого примера, собственные значения в обоих случаях вычислены правильно, но нормы невязок различаются очень существенно, что свидетельствует о том, что собственные векторы в первом случае вычислены неверно.

Обобщенная проблема собственных значений состоит в нахождении нетривиальных решений системы уравнений

$Ar = \lambda Br$,

где A, B - квадратные матрицы порядка n;

r - вектор-столбец размера 1×n, называемый *обобщенным собственным вектором;*

λ - скаляр, называемый *обобщенным собственным значением.*

Вычисления с использованием комплексных матриц:

```
A =                                R =                                          D =
 2.0000  -2.0000  -0.6667   0.5747 +0.4598i  0.5747  -0.4598i  0.8827     1.7903 +1.7508i  0          0
 0.6667   2.0000  -2.0000   0.4848  -0.3522i  0.4848  +0.3522i -0.3044    0          1.7903 -1.7508i  0
 0.4000   0.6667   2.0000  -0.0659  -0.3080i -0.0659  +0.3080i  0.3580    0          0        2.4193

A' =                               L =                                          D1 =
 2.0000   0.6667   0.4000   0.5748 +0.4597i  0.5748  -0.4597i  0.8827     1.7903 +1.7508i  0          0
-2.0000   2.0000   0.6667   0.4847  -0.3524i  0.4847  +0.3524i -0.3044    0          1.7903 -1.7508i  0
-0.6667  -2.0000   2.0000  -0.0660  -0.3080i -0.0660  +0.3080i  0.3581    0          0        2.4193

L1 =                                                   L1' * A * R =
 -0.6802 +0.5849i  -0.6802  -0.5849i  1.2179            1.7903 +1.7508i  0          0
 -0.6057  -0.4081i  -0.6057  +0.4081i  -0.4200           0          1.7903 -1.7508i  0
  0.0670  -0.3780i   0.0670  +0.3780i  0.4940            0          0        2.4193

                                                       R'*A'*L1 =
                                                       1.7903-1.7508i  0.0000 +0.0000i0.0000
                                                       0.0000-0.0000i  1.7903 +1.7508i0.0000
                                                       0.0000+0.0000i  0.0000  -0.0000i2.4193
```

Вычисления с использованием только действительных матриц:

```
A =                          R =                          D =
 2.0000  -2.0000  -0.6667    -0.2517   0.1893   0.3581     1.7904   1.7508   0
 0.6667   2.0000  -2.0000    -0.5234  -0.2917  -0.3044    -1.7508   1.7904   0
 0.4000   0.6667   2.0000     0.1756  -0.7148   0.8827     0        0        2.4193
A' =                         L =                          D1 =
 2.0000   0.6667   0.4000     0.5748   0.4597   0.8827     1.7904   1.7508   0
-2.0000   2.0000   0.6667     0.4847  -0.3524  -0.3044    -1.7508   1.7904   0
-0.6667  -2.0000   2.0000    -0.0660  -0.3080   0.3581     0        0        2.4193
                             L1 =                         L1' * A * R =
                             -1.3603   1.1698   1.2179     1.7904   1.7508   0
                             -1.2114  -0.8162  -0.4200    -1.7508   1.7904   0
                              0.1341  -0.7560   0.4940     0        0        2.4193
                                                          R' * A' * L1 =
                                                           1.7904  -1.7508   0.0000
                                                           1.7508   1.7904   0.0000
                                                           0.0000   0.0000   2.4193
```

Если B - невырожденная матрица, то система может быть рассмотрена как алгебраический эквивалент системы обыкновенных дифференциальных уравнений в неявной форме Коши, а задача может быть сведена к обычной проблеме собственных значений

$B^{-1}Ar = \lambda r$.

В случае, когда B - вырожденная матрица, система уравнений представляет собой смешанную систему дифференциальных и алгебраических уравнений и для ее решения необходимо применять специальные методы.

Функция d = eig(A, B) вычисляет обобщенные собственные значения матрицы A.

Функция [R, D] = eig(A, B) вычисляет диагональную матрицу D обобщенных собственных значений и матрицу R правых обобщенных собственных векторов, удовлетворяющих соотношению A * R = B * R * D. Эти векторы нормированы так, что норма каждого из них равна единице.

Алгоритм:

Для действительных матриц функция eig(A) использует следующие модули пакета EISPACK [1,2]: balance, balbak, orthes, ortran и hqr2. Модули balance и balbak связаны с операциями масштабирования и восстановления; модуль orthes осуществляет приведение матрицы к форме Хессенберга посредством ортогональных подобных преобразований; модуль ortran запоминает все преобразования; модуль hqr2 вычисляет собственные значения и векторы матрицы в верхней форме Хессенберга с использованием QR-алгоритма Франсиса и Кублановской [3].

Функция eig(A, B) использует другие модули пакета EISPACK [1,2]: qzhes, qzit, qzval и qzvec, основанные на QZ-алгоритме.

Для комплексных матриц функция eig(A) использует QZ-алгоритм, решая задачу в форме eig(A, eye(A)).

Диагностические сообщения:

Если в течение 30*n итераций собственные значения не найдены, выдается сообщение

Solution will not converge.

Решение не сходится.

Сопутствующие функции: BALANCE, CONDEIG, HESS, QZ, SCHUR.

Ссылки:

1. Smith B. T., Boyle J. M., Dongarra J. J., Garbow B. S., Ikebe Y., Klema V., Moler C. B. Matrix Eigensystem Routines - EISPACK//Guide. Lecture Notes in Computer Science. Vol. 6. Berlin, 1976.
2. Garbow B. S., Boyle J. M., Dongarra J. J., Moler C. B.. Matrix Eigensystem Routines - EISPACK Guide Extension//Lecture Notes in Computer Science. Vol. 51. Berlin, 1977.
3. Уилкинсон, Райнш. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра: Пер. с англ. М.: Машиностроение, 1976. 390 с.
4. Moler C. B., Stewart G. W. An Algorithm for Generalized Matrix Eigenvalue Problems//SIAM J. Numer. Anal. 1973. Vol. 10. N 2.

CONDEIG — Число обусловленности задачи на собственные значения

Синтаксис:

```
      s = condeig(A)
[V, D, s] = condeig(A)
```

Описание:

Функция s = condeig(A) вычисляет вектор чисел обусловленности задачи на собственные значения для матрицы A. Эти числа определяются как обратные значения косинусов углов между левыми и правыми собственными векторами, соответствующими некоторому собственному значению.

Функция [V, D, s] = condeig(A) эквивалентна последовательности двух операторов [V, D] = eig(A); s = condeig(A);.

Большие по величине числа обусловленности означают, что матрица A имеет собственные значения, близкие к кратным.

Сопутствующие функции: BALANCE, COND, EIG.

BALANCE **Масштабирование матрицы**

Синтаксис:

B = balance(A)
[D, B] = balance(A)

Описание:

Несимметрические матрицы могут проявлять плохую обусловленность при вычислении их собственных значений. Малые возмущения элементов матрицы, такие, как ошибки округления, могут вызывать значительные погрешности в собственных значениях. Величина, связывающая погрешность вычисления собственных значений с погрешностью исходных данных, называется *числом обусловленности задачи на собственные значения* и вычисляется следующим образом:

k = cond(R) = norm(R) * norm(inv(R)),

где [R, D] = eig(A).

Цель масштабирования - перевести плохую обусловленность матрицы собственных векторов в диагональное масштабирование. Масштабирование не может превратить несимметрическую матрицу в симметрическую, но делается попытка выравнять нормы строк и соответствующих столбцов. Поскольку масштабирование реализуется введением множителей, которые являются степенями основания 2, то никаких ошибок округления при этом не привносится.

Функция B = balance(A) возвращает масштабированную матрицу.

Функция [D, B] = balance(A) кроме масштабированной матрицы возвращает также диагональную матрицу D, элементы которой являются степенями основания 2; матрица B - это результат преобразования подобия

B = D \ A * D.

Функция eig(A) автоматически масштабирует A перед вычислением собственных значений D. Масштабирование можно подавить, если использовать обращение вида eig(A, 'nobalance').

Пример:

Рассмотрим матрицу порядка 3, которая имеет большой разброс значений элементов - большие наддиагональные и малые поддиагональные элементы.

```
A =      1       100     10000
         0.01      1       100
         0.0001    0.01      1
```

Вычислим собственные значения и векторы матрицы A.

```
[RA, DA] = eig(A)
A =                          RA =                          DA =
 1.0e+004*
0.0001 0.0100 1.0000         0.9999  -1.0000  -0.9998     3.0000 0       0
0.0000 0.0001 0.0100         0.0100   0.0054   0.0209     0      0.0000 0
0.0000 0.0000 0.0001         0.0001   0.0000  -0.0001     0      0      0.0000
                             cond(RA) = 1.4073e+004
```

Матрица собственных векторов плохо обусловлена.

Выполним масштабирование матрицы A.

```
[D, B] = balance(A)
A =                          D =                          B =
 1.0e+004 *                   1.0e+003 *
  0.0001        0.0100         2.0480  0       0          1.0000        1.5625
1.0000                         0       0.0320  0         1.2207
  0.0000        0.0001         0       0       0.0003     0.6400        1.0000
0.0100                       cond(D) = 8192              0.7813
  0.0000        0.0000                                     0.8192        1.2800
0.0001                                                    1.0000
```

После масштабирования значения элементов матрицы B оказались выравненными по величине.

Вычислим собственные значения и векторы матрицы B.

```
[RB, DB] = eig(B)
B =                          RB =                          DB =
 1.0000  1.5625  1.2207       0.6933     -0.8903     -     3.0000  0       0
 0.6400  1.0000  0.7813      0.5274                       0       0.0000  0
 0.8192  1.2800  1.0000       0.4437          0.3070      0       0       0.0000
                             0.7064
                               0.5679      0.3364     -
                             0.4721
                             cond(RB) = 1.9381
```

Матрица собственных векторов очень хорошо обусловлена. Плохая обусловленность сконцентрирована в диагональной матрице масштабирования D.

Алгоритм:

Функция balance является встроенной функцией интерпретатора MATLAB; она использует алгоритмы, первоначально написанные на языке АЛГОЛ [1], а затем реализованные на языке Fortran в составе пакета EISPACK: balance, balbak [2].

Ограничения:

Обычно масштабирование улучшает обусловленность матрицы, гарантируя большую точность вычислений. Однако когда матрица содержит очень малые по величине элементы, которые находятся в пределах ошибок округления, масштабирование может сделать их сравнимыми с другими элементами матрицы, что в состоянии привести к неправильным результатам.

Диагностические сообщения:

Если матрица не квадратная, выдается сообщение

Matrix must be square.

Матрица должна быть квадратной.

Сопутствующие функции: EIG, HESS, SCHUR.

Ссылки:

1. Уилкинсон, Райнш. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра: Пер. с англ. М.: Машиностроение, 1976. 390 с.
2. Garbow B. S., Boyle J. M., Dongarra J. J., Moler C. B. Matrix Eigensystem Routines - EISPACK Guide Extension//Lecture Notes in Computer Science. Vol. 51. Berlin, 1977.

PLANEROT — Преобразование Гивенса

Синтаксис:

[G, y] = planerot(x)

Описание:

Функция [G, y] = planerot(x), где x - вектор-столбец из двух компонентов, возвращает ортогональную матрицу G порядка 2, такую, что выполняется условие y = Gx и y(2) = 0.

Преобразование Гивенса применяется для исключения элементов матрицы с целью ее приведения к более простой форме (Хессенберга, трехдиагональной и т. п.).

Сопутствующие функции: QRDELETE, QRINSERT.

HESS — Приведение к форме Хессенберга

Синтаксис:

H = hess(A)

[P, H] = hess(A)

Описание:

Функция H = hess(A) возвращает матрицу в верхней форме Хессенберга, элементы которой hij с номерами i > j + 1, то есть расположенные ниже первой поддиагонали, равны нулю. Если матрица A симметрическая или эрмитова, то матрица Хессенберга вырождается в трехдиагональную.

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

A = P * H * P, P' * P = eye(size(A)).

Пример:

Рассмотрим приведение матрицы A = magic(5) размером 5×5 к верхней форме Хессенберга.

```
A = magic(5), H = hess(A)
A =                          H =
 17  24   1   8  15            17.0000  -28.9413    1.8470  -4.4603   2.2572
 23   5   7  14  16           -27.6767   33.9399   26.1875  -2.2280   1.2675
  4   6  13  20  22                 0   25.0964   20.6871  -6.6055  -0.1973
 10  12  19  21   3                 0         0   -5.9630 -16.8163 -12.4454
 11  18  25   2   9                 0         0         0  -9.0122  10.1893
```

Алгоритм:

Для действительных матриц функция hess(A) использует следующие модули пакета EISPACK [1-2]: ortran и orthes. Модуль orthes осуществляет приведение матрицы к верхней форме Хессенберга посредством ортогональных подобных преобразований; модуль ortran запоминает все преобразования.

Для комплексных матриц функция hess(A) использует модуль qzhes пакета EISPACK.

Сопутствующие функции: EIG, QZ, SCHUR.

Ссылки:

1. Smith B. T., Boyle J. M., Dongarra J. J., Garbow B. S., Ikebe Y., Klema V., Moler C. B.. Matrix Eigensystem Routines - EISPACK Guide//Lecture Notes in Computer Science. Vol. 6. Berlin, 1976.
2. Garbow B. S., Boyle J. M., Dongarra J. J., Moler C. B.. Matrix Eigensystem Routines - EISPACK Guide Extension//Lecture Notes in Computer Science. Vol. 51. Berlin, 1977.

SCHUR, RSF2CSF — Приведение к форме Шура

Синтаксис:

T = schur(A)
[U, T] = schur(A)
[U, T] = rsf2csf(U,T)

Описание:

Функция T = schur(A) возвращает матрицу в форме Шура. *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 * H * U', U' * U = eye(size(A)).

Если исходная матрица A действительная, то возвращается *действительная форма Шура*, если комплексная, то *комплексная форма Шура*. Функции cdf2rdf и rsf2csf обеспечивают преобразование из одной формы в другую.

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

Пример:

Рассмотрим приведение тестовой матрицы с двумя кратными комплексными и одним действительным собственным значениями к действительной форме Шура [1].

```
A =
    15   11    6   -9  -15
     1    3    9   -3   -8
     7    6    6   -3  -11
     7    7    5   -3  -11
    17   12    5  -10  -16
[U, T] = schur(A)
U =                                                T =
  0.2868  0.0207  0.7227  0.2502 -0.5766   -1.0000  20.1766   4.8432  33.1468 -15.2146
  0.4854  0.4640 -0.5919  0.1372 -0.4241         0  -0.0399  -2.0829  -2.2176   8.8043
  0.4192  0.0148  0.0802  0.6744  0.6023         0   7.2596   3.0399  10.1089 -10.9556
  0.3530  0.5466  0.3290 -0.5861  0.3533         0        0        0   0.7171   2.7149
  0.6178 -0.6966 -0.1129 -0.3467 -0.0096         0        0        0  -4.9221   2.2829
```

Преобразуем действительную квазитреугольную форму Шура в комплексную треугольную.

```
[U1, T1] = rsf2csf(U, T)
U1 =
  0.2868  0.6332 + 0.0090i  -0.1534 - 0.3133i  0.4310 + 0.1457i  0.2745 + 0.3358i
  0.4854 -0.6085 + 0.2012i  -0.2984 + 0.2566i  0.3230 + 0.0799i  0.1643 + 0.2470i
  0.4192  0.0680 + 0.0064i  -0.0280 - 0.0348i -0.5696 + 0.3928i  0.4645 - 0.3508i
  0.3530  0.1878 + 0.2370i  -0.5434 - 0.1426i -0.2088 - 0.3413i -0.5156 - 0.2058i
  0.6178  0.0307 - 0.3020i   0.6352 + 0.0490i  0.0520 - 0.2019i -0.2771 + 0.0056i
T1 =
 -1.0000  0.4967 + 8.7481i  -18.6913 - 2.0999i  7.9820 +19.3046i  28.5534 + 8.8610i
        0  1.5000 + 3.5707i   -5.8300 - 1.5148i  7.3239 + 8.3731i   6.7070 + 7.8425i
        0                 0   1.5000 - 3.5707i  2.0252 + 3.2914i  -1.9851 + 7.4522i
        0                 0                  0  1.5000 + 3.5707i   2.4562 + 1.1359i
        0                 0                  0                 0   1.5000 - 3.5707i
```

Алгоритм:

Для действительных матриц функция schur(A) использует следующие модули пакета EISPACK [2,3]: ortran, orthes и hqr2. Модуль orthes осуществляет приведение матрицы к верхней форме Хессенберга посредством ортогональных подобных преобразований; модуль ortran запоминает все преобразования. Модуль hqr2 вычисляет собственные значения матрицы в верхней форме Хессенберга на основе QR-алгоритма Франсиса - Кублановской.

Для комплексных матриц функция schur(A) использует модули qzhes, qzit, qzval и qzvec пакета EISPACK.

Сопутствующие функции: HESS, EIG, QZ.

Ссылки:

1. Уилкинсон, Райнш. *Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра:* Пер. с англ. М.: Машиностроение, 1976. 390 с.
2. Smith B. T., Boyle J. M., Dongarra J. J., Garbow B. S., Ikebe Y., Klema V., Moler C. B. Matrix Eigensystem Routines - EISPACK Guide//Lecture Notes in Computer Science. Vol. 6. Berlin, 1976.
3. Garbow B. S., Boyle J. M., Dongarra J. J., Moler C. B. Matrix Eigensystem Routines - EISPACK Guide Extension//Lecture Notes in Computer Science. Vol. 51. Berlin, 1977.

QZ — Приведение пары матриц к обобщенной форме Шура

Синтаксис:

[AA, BB, Q, Z, V] = qz(A, B)

Описание:

Многие задачи линейной алгебры - решение матричных уравнений Сильвестра и Риккати, смешанных систем дифференциальных и линейных алгебраических уравнений - приводят к необходимости одновременного приведения пары матриц к форме Шура.

Функция [AA, BB, Q, Z, V] = qz(A, B) возвращает комплексные верхние треугольные матрицы AA и BB, соответствующие матрицы приведения Q и Z, а также вектор обобщенных собственных векторов V, так что

Q * A * Z = AA;
Q * B * Z = BB.

Обобщенные собственные значения могут быть найдены исходя из следующего условия:

A * V * diag(BB) = B * V * diag(AA).

Пример:

Для системы обыкновенных дифференциальных уравнений в неявной форме Коши с одним входом и одним выходом

$$Q\dot{x} + Rx = bu;$$
$$y = c^T x + du$$

задачи вычисления полюсов и нулей соответствующей передаточной функции определяются следующим образом [1]:

- вычисление полюсов:

  $Rr = -\lambda Qr;$

- вычисление нулей:

$$\begin{bmatrix} -R & b \\ c^T & d \end{bmatrix} r = \lambda \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} r.$$

Нетрудно видеть, что обе задачи требуют решения обобщенной проблемы собственных значений.

Описание системы:

```
Q =                          R =                          b =
     1.0000   0                   1.1190  -1.0000            31.0960
     0.1920   1.0000             36.4800   1.5380             0.1284
C = [ 0.6299   0 ]           d = -0.0723
```

Расчет полюсов:

```
[AA, BB] = qz(R, -Q)
AA =                                BB =                                    diag(AA)./diag(BB) =
5.5039+ 2.7975i  24.8121 - 25.3646i   -0.6457 + 0.7622i  -0.1337 + 0.1378i    -1.4245 - 6.0143i
0.0000 + 0.0000i  5.5158 - 2.8036i          0            -0.6471 - 0.7638i    -1.4245 + 6.0143i
```

Расчет нулей:

```
A = [-R   b                          B = [      -Q          zeros(size(b))
      C   d]                                zeros(size(C))       0          ]
   -1.1190  1.0000   0.1284                        -1.0000  0        0
  -36.4800 -1.5380  31.0960                        -0.1920 -1.0000   0
   0.6299   0        0.0723                         0       0        0
[AA, BB] = qz(A, B)
AA =                          BB =                                diag(AA)./diag(BB) =
 31.0963  -0.7165  -36.5109    0      0.9860  -0.2574              Inf
  0.0000   1.0647    0.9229    0      0.0657   0.9964              16.2035
  0        0.0000    0.5119    0      0       -0.0354             -14.4735
```

Алгоритм:

Средством вычисления формы Шура для пары матриц {A, B} является созданный Моулером и Стьюартом QZ-алгоритм [2,3], который реализован в рамках пакета EISPACK [4] и использует модули qzhes, qzit, qzval и qzvec.

Сопутствующие функции: EIG.

Ссылки:

1. Потемкин В. Г., Кутузова Г. Н. Особенности исследования дискретно-непрерывных комплексов на ЭВМ: *Учеб. пособие*. М.: МИФИ, 1988. 56 с.
2. Икрамов Х. Д. *Численное решение матричных уравнений*. М.: Наука, 1984. 192 с.
3. Moler C. B., Stewart G. W. An Algorithm for Generalized Matrix Eigenvalue Problems//*SIAM J. Numer. Anal.* 1973. Vol. 10. N 2.
4. Garbow B. S., Boyle J. M., Dongarra J. J., Moler C. B. Matrix Eigensystem Routines - EISPACK Guide Extension//Lecture Notes in Computer Science. Vol. 51. Berlin, 1977.

POLYEIG **Вычисление собственных значений матричного полинома**

Синтаксис:

[R, d] = polyeig(A0, A1, ..., Ap)

Описание:

Функция [R, d] = polyeig(A0, A1, ..., Ap) решает полную проблему собственных значений для матричного полинома степени p вида

$(A_0 + \lambda * A1 + \ldots + \lambda^p * A_p) * r = 0.$

Входными переменными этой функции являются p+1 квадратная матрица $A_0, A_1, \ldots, A_p$ порядка n. Выходными переменными - матрица собственных векторов R размера n×(n×p) и вектор собственных значений d длины n×p.

Для некоторых значений p и n функция polyeig становится равносильной другим функциям системы MATLAB:

- p = 0, функция polyeig(A) равносильна функции eig(A);
- p = 1, функция polyeig(A, B) равносильна функции eig(A, -B);
- n = 1, функция polyeig(a0, a1, ..., ap) для скаляров a0, ..., ap, равносильна функции roots([ap .. a1 a0]).

Алгоритм:

Задача сводится к решению обобщенной проблемы собственных значений для пары матриц A и B порядка n × p. В частном случае, когда p = 4, эти матрицы имеют вид

$$A = \begin{bmatrix} A_0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}; \quad B = \begin{bmatrix} -A_1 & -A_2 & -A_3 & -A_4 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix};$$

Если одна (но не обе) из матриц A_0 или A_p вырожденна, то некоторые из собственных значений могут оказаться равными нулю или Inf.

Если обе матрицы A_0 и A_p вырожденны, то задача оказывается плохо обусловленной. С точки зрения теории это означает, что решения может не существовать или оно может быть неединственным. С вычислительной точки зрения решение может оказаться неточным.

В алгоритме сделана попытка выявить эту ситуацию и сформировать соответствующее предупреждение.

Диагностические сообщения:

Если обе матрицы A_0 и A_p близки к вырожденным, выдается сообщение

Warning: Rank deficient generalized eigenvalue problem.

Eigenvalues are not well determined. Results may be inaccurate.

Предупреждение: Неполный ранг для обобщенной проблемы.

Собственные значения плохо обусловлены. Результат может быть неточным.

Сопутствующие функции: EIG, QZ.

SVD **Сингулярное разложение матрицы**

Синтаксис:

s = svd(A)
[U, S, V] = svd(A)
[U, S, V] = svd(A, 0)

Описание:

Если A - действительная матрица размера m×n (m ≥ n), то ее можно представить в виде [1]:

$A = U * S * V^T$,

где $U^TU = V * V^T = I_n$ и $S = diag(s_1, ...s_n)$.

Такое разложение называется *сингулярным разложением матрицы* A.

Матрица U сформирована из n ортонормированных собственных векторов, соответствующих n наибольшим собственным значениям матрицы AA^T, а матрица V - из ортонормированных собственных векторов матрицы A^TA. Диагональные элементы матрицы S - неотрицательные значения квадратных корней из собственных значений матрицы A^TA; они называются *сингулярными числами.*

Допустим, что $s_1 \geq s_2 \geq ... \geq s_n \geq 0$. Если ранг матрицы A равен r, то значения $s_{r+1} = s_{r+2} = ... = s_n = 0$.

Существует другое, более экономное сингулярное разложение:

$A = U_r * S_r * V_r^T$,

где $U_r^TU_r = V_r * V_r^T = I_r$ и $S_r = diag(s_1, ..., s_r)$.

Функция s = svd(A) вычисляет только сингулярные числа матрицы A.

Функция [U, S, V] = svd(A) вычисляет диагональную матрицу S тех же размеров, которые имеет и матрица A с неотрицательными диагональными элементами в порядке их убывания, а также унитарные матрицы преобразований U и V.

Функция [U, S, V] = svd(A, 0) выполняет экономное сингулярное разложение.

Алгоритм:

Функция svd(A) использует модуль svd пакета LINPACK [4].

Диагностические сообщения:

Если в течение 75 итераций QR-преобразования сингулярные значения не найдены, выдается сообщение

Solution will not converge.

Решение не сходится.

Пример 1:

Рассмотрим прямоугольную матрицу размера 4×2.

```
A =
   1   2
   3   4
   5   6
   7   8
```

Полное сингулярное разложение

```
[U, S, V] = svd(A)

U =                                    V =                 S =
  0.1525  0.8226 -0.3945 -0.3800         0.6414 -0.7672      14.2691  0
  0.3499  0.4214  0.2428  0.8007         0.7672  0.6414      0        0.6268
  0.5474  0.0201  0.6979 -0.4614                             0        0
  0.7448 -0.3812 -0.5462  0.0407                             0        0
```

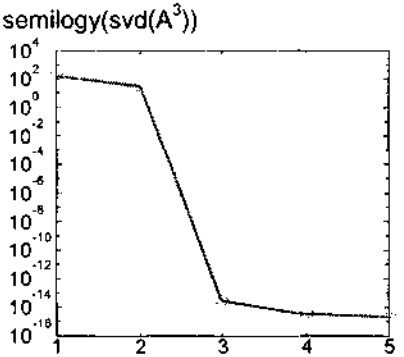
Экономное сингулярное разложение

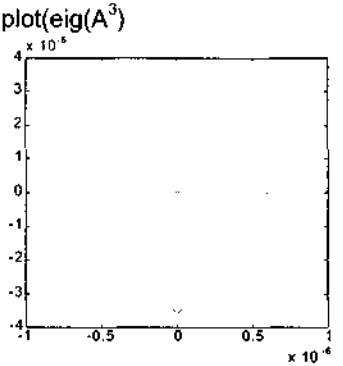
```
[U, S, V] = svd(A, 0)

U =                    V =                 S =
  0.1525  0.8226         0.6414 -0.7672      14.2691  0
  0.3499  0.4214         0.7672  0.6414      0        0.6268
  0.5474  0.0201
  0.7448 -0.3812
```

Пример 2:

Рассмотрим матрицу порядка 5, которая в пределах ошибок округления имеет только нулевые собственные значения [2].

```
A = chebspec(5)

A =        5.5000 -6.8284  2.0000 -1.1716  0.5000
           1.7071 -0.7071 -1.4142  0.7071 -0.2929
          -0.5000  1.4142  0.0000 -1.4142  0.5000
           0.2929 -0.7071  1.4142  0.7071 -1.7071
          -0.5000  1.1716 -2.0000  6.8284 -5.5000

[U, S, V] = svd(A^3)

U =
   0.5774 -0.4472 -0.5412  0.2121  0.3588
   0.4082 -0.4472  0.2245 -0.2794 -0.7106
   0.0000 -0.4472  0.6753 -0.1695  0.5614
  -0.4082 -0.4472  0.0819  0.7588 -0.2256
  -0.5774 -0.4472 -0.4404 -0.5220  0.0159

S =
   155.5378  0         0         0         0
   0         32.8634   0         0         0
   0         0         0.0000    0         0
   0         0         0         0.0000    0
   0         0         0         0         0.0000

V =
   0.2673 -0.4082  0.6267 -0.5498  0.2588
  -0.5345  0.5774  0.0315 -0.4589  0.4115
   0.5345  0.0000 -0.5883 -0.1319  0.5923
  -0.5345 -0.5774 -0.0528  0.3475  0.5073
   0.2673  0.4082  0.5074  0.5908  0.3943
```





Из анализа графиков следует, что 3 сингулярных числа имеют значения меньше 10^{-14} и существенно отличаются от остальных; собственные значения находятся в круге радиусом $0.5 * 10^{-6}$, то есть являются кратными. Дефект матрицы A^3 равен 3.

В рассматриваемом случае это означает, что имеется 3 клетки Жордана, порядок которых пока неизвестен. Применяя пакет программ JORD [3], можно установить, что в данном случае имеется одна клетка 1-го и две клетки 2-го порядка.

Сопутствующие функции: GSVD, SVDS.

Ссылки:

1. Уилкинсон, Райнш. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра: Пер. с англ. М.: Машиностроение, 1976. 390 с.
2. Higham N. J. The Test Matrix Toolbox for MATLAB (version 3.0)//Numerical Analysis Report. Vol. 276. Manchester, 1995.
3. Потемкин В. Г. *Пакет программ JORD*. М.: МИФИ, 1995.
4. Dongarra J. J., Bunch J. R., Moler C. B., Stewart G. W. LINPACK User's Guide. Philadelphia, 1979.

GSVD — Обобщенное сингулярное разложение матрицы

Синтаксис:

```
[U, V, X, C, S] = gsvd(A, B)
[U, V, X, C, S] = gsvd(A, B, 0)
        sigma = gsvd(A,B)
```

Описание:

Функция [U, V, X, C, S] = gsvd(A, B) возвращает унитарные матрицы U и V, квадратную (как правило) матрицу X и неотрицательные диагональные матрицы C и S, такие, что справедливы следующие соотношения:

```
A = U*C*X';
B = V*S*X';
C'*C + S'*S = I.
```

Матрицы A и B должны иметь одинаковое количество столбцов, но могут иметь разное количество строк; если A - матрица размера m×p, а B - размера n×p, то U - размера m×m, V - размера n×n, X - размера p×q, где q = min(m+n, p). Ненулевые элементы S всегда размещены на главной диагонали. Если m ≥ p, то ненулевые элементы C также размещены на главной диагонали; если m < p, то ненулевые элементы размещаются на диагонали diag(C, p-m). Это позволяет упорядочить диагональные элементы таким образом, чтобы сингулярные числа располагались в неубывающем порядке.

Функция [U, V, X, C, S] = gsvd(A, B, 0) реализуст экономичное разложение, когда выходные матрицы U и V содержат самое большее p столбцов, а матрицы C и S - самое большее p строк. Обобщенные сингулярные значения при этом определяются как diag(C)./diag(S).

Функция sigma = gsvd(A, B) возвращает вектор обобщенных сингулярных значений sqrt(diag(C'*C)./diag(S'*S)).

Когда B квадратная несингулярная матрица, ее обобщенные сингулярные значения совпадают с обычными сингулярными значениями svd(A/B), но отсортированными в обратном порядке; обратные значения могут быть получены с помощью функции gsvd(B, A).

В принятой формулировке GSVD-разложения не делается никаких предположений относительно рангов матриц A и B. Матрица X имеет полный ранг, если и только если матрица [A; B] имеет полный ранг. Фактически это означает, что функции svd(X) и cond(X) равны соответственно svd([A; B]) и cond([A; B]).

Другие формулировки, использованные, например, в работе [1], требуют, чтобы нуль-пространства null(A) и null(B) не пересекались, и поэтому в них X заменяется на inv(X) или inv(X').

Заметим, что когда нуль-пространства null(A) и null(B) пересекаются, то ненулевые элементы матриц C и S не могут быть определены однозначно.

Алгоритм:

Алгоритм обобщенного сингулярного разложения использует C-S разложение, описанное в работе **[1]**, а также встроенные функции svd и qr. C-S разложение в виде подфункции M-файла gsvd.

Пример 1:

```
A = reshape(1:15, 5, 3)
A =
    1    6   11
    2    7   12
    3    8   13
    4    9   14
    5   10   15
B = magic(3)
B =
    8    1    6
    3    5    7
    4    9    2
```

```
[U, V, X, C, S] = gsvd(A, B)
U =
   0.6303  -0.6457  -0.4279   0.0183   0.0493
  -0.6756  -0.3296  -0.4375   0.1160   0.4797
  -0.2512  -0.0135  -0.4470  -0.5838  -0.6294
   0.0080   0.3026  -0.4566   0.7465  -0.3778
   0.2885   0.6187  -0.4661  -0.2970   0.4781
V =
  -0.7071   0.6946   0.1325
   0.0000   0.1874  -0.9823
   0.7071   0.6946   0.1325
X =
  -2.8284   9.3761  -6.9346
   5.6569   8.3071 -18.3301
  -2.8284   7.2381 -29.7256
C =
   0.0000        0        0
        0   0.3155        0
        0        0   0.9807
        0        0        0
        0        0        0
S =
   1.0000        0        0
        0   0.9489        0
        0        0   0.1957
```

GSVD-разложение формирует ортогональные матрицы U и V размеров 5×5 и 3×3, соответственно, а также невырожденную матрицу X размера 3×3 и матрицы C и S размеров 5×3 и 3×3, соответственно. Поскольку матрица A сингулярная, первый диагональный элемент матрицы C равен 0.

Пример 2:

Экономичное GSVD-разложение

```
[U, V, X, C, S] = gsvd(A, B, 0)
U =
   0.1294  -0.6457  -0.4279
  -0.6459  -0.3296  -0.4375
   0.7311  -0.0135  -0.4470
  -0.0420   0.3026  -0.4566
  -0.1726   0.6187  -0.4661
V =
  -0.7071   0.6946   0.1325
  -0.0000   0.1874  -0.9823
   0.7071   0.6946   0.1325
X =
  -2.8284   9.3761  -6.9346
   5.6569   8.3071 -18.3301
  -2.8284   7.2381 -29.7256
```

```
C =
   0.0000        0        0
        0   0.3155        0
        0        0   0.9807
S =
   1.0000        0        0
        0   0.9489        0
        0        0   0.1957
```

формирует ортогональные матрицы U и V размеров 5×3 и 3×3, соответственно, а также невырожденную матрицу X размера 3×3 и матрицы C и S размеров 3×3 и 3×3, соответственно.

Пример 3:

Вычисление обобщенных сингулярных чисел:

```
sigma = gsvd(A, B)
sigma =
   0.0000
   0.3325
   5.0123
```

Сравним с обычными сингулярными числами

```
svd(A/B)
ans =
   5.0123
   0.3325
   0.0000
```

Как уже описывалось ранее, они имеют обратное расположение.

Сопутствующие функции: SVD.

Ссылки:

1. Golub G.H., Van Loan C. F. Matrix Computations. 3rd ed. Baltimore: Johns Hopkins University Press, 1996.

Вычисление функций от матриц

EXPM, EXPM1, EXPM2, EXPM3 — **Вычисление матричной экспоненты**

Синтаксис:

```
Y = expm(A)        Y = expm2(A)
Y = expm1(A)       Y = expm3(A)
```

Описание:

Функция Y = expm(A) является встроенной функцией интерпретатора системы MATLAB и вычисляет функцию e^A от матрицы A.

Функция Y = expm1(A) является M-файлом, который полностью соответствует встроенной функции expm(A). Он вычисляет функцию e^A, используя разложение Паде матрицы A [1].

Функция Y = expm2(A) вычисляет функцию e^A, используя разложение Тейлора матрицы A [2]. Этот метод имеет меньшую скорость сходимости по сравнению с разложением Паде.

Функция Y = expm3(A) вычисляет функцию e^A, используя спектральное разложение матрицы A:

```
[R, D] = eig(A);
Y = R * diag(exp(diag(D))) / R;,
```

которое, строго говоря, справедливо только для случая различных собственных значений.

Замечание:

Функцию матричной экспоненты expm(A) не следует путать с функцией exp(A), которая вычисляет экспоненту от каждого элемента массива A.

Пример:

Рассмотрим тестовую матрицу порядка n = 5 с дефектом 1, имеющую 5 кратных собственных значений, равных нулю [3].

A = chebspec(5)					Y = expm(A)				
5.5000	-6.8284	2.0000	-1.1716	0.5000	21.0000	-32.4853	21.0000	-15.5147	7.0000
1.7071	-0.7071	-1.4142	0.7071	-0.2929	11.1569	-15.7782	9.2426	-6.5355	2.9142
-0.5000	1.4142	0.0000	-1.4142	0.5000	1.0000	0.0000	0.0000	0.0000	0.0000
0.2929	-0.7071	1.4142	0.7071	-1.7071	-0.1569	0.5355	0.7574	-0.2218	0.0858
-0.5000	1.1716	-2.0000	6.8284	-5.5000	0.0000	0.0000	1.0000	0.0000	0.0000

Y1 = expm1(A)					Y2 = expm2(A)				
21.0000	-32.4853	21.0000	-15.5147	7.0000	21.0000	-32.4853	21.0000	-15.5147	7.0000
11.1569	-15.7782	9.2426	-6.5355	2.9142	11.1569	-15.7782	9.2426	-6.5355	2.9142
1.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000
-0.1569	0.5355	0.7574	-0.2218	0.0858	-0.1569	0.5355	0.7574	-0.2218	0.0858
0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000

Y3 = expm3(A)				
21.0015+0.0000i	-32.4884-0.0001i	21.0031+0.0001i	-15.5178-0.0001i	7.0015+0.0000i
11.1577+0.0000i	-15.7798-0.0000i	9.2443+0.0000i	-6.5372-0.0000i	2.9150+0.0000i
0.9999+0.0000i	0.0002-0.0000i	-0.0002+0.0000i	0.0002-0.0000i	-0.0001+0.0000i
-0.1569-0.0000i	0.5356+0.0000i	0.7572-0.0000i	-0.2217+0.0000i	0.0857-0.0000i
0.0000-0.0000i	0.0001+0.0000i	0.9999-0.0000i	0.0001+0.0000i	0.0000-0.0000i

norm(Y1 - Y)	norm(Y1 - Y2)	norm(Y1 - Y3)
0	1.0560e-013	0.0065

Как следует из анализа полученных данных, функции expm(A) и expm1(A) дают совпадающие результаты, функция expm2(A) имеет погрешность в пределах ошибки округления, однако функция expm3(A) имеет ошибки в третьем знаке, что является следствием того, что исходная система имеет кратные собственные значения.

Используя пакет программ JORD [4], можно выявить точную структуру формы Жордана матрицы A:

R =

0.2342	0.2342	0.0679	-0.0102	-0.0050
0.2342	0.1656	0.0093	-0.0210	0.0000
0.2342	0.0000	-0.0492	0.0000	0.0099
0.2342	-0.1656	0.0093	0.0210	0.0000
0.2342	-0.2342	0.0679	0.0102	-0.0050

J =

0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1
0	0	0	0	0

В данном случае это 1 клетка Жордана порядка 5, характеризующая простую однократную вырожденность.

Аналитическая функция f(J), где J - клетка Жордана, соответствующая собственному значению λ, может быть вычислена следующим образом [4]:

$$f(J) = \begin{bmatrix} f(\lambda) & \frac{f'(\lambda)}{1!} & \frac{f''(\lambda)}{2!} & \dots & \frac{f^{(k)}(\lambda)}{k!} \\ 0 & f(\lambda) & \frac{f'(\lambda)}{1!} & \dots & \frac{f^{(k-1)}(\lambda)}{(k-1)!} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \frac{f'(\lambda)}{1!} \\ 0 & 0 & 0 & \dots & f(\lambda) \end{bmatrix}.$$

В рассматриваемом случае матрица e^J равна

EJ =

1.0000	1.0000	0.5000	0.1667	0.0417
0	1.0000	1.0000	0.5000	0.1667
0	0	1.0000	1.0000	0.5000
0	0	0	1.0000	1.0000
0	0	0	0	1.0000

,

а матрица $Y0 = e^A = R * e^J * R^{-1}$ может быть вычислена так:

Y0 = R * EJ / R.

Вычисленная матрица Y0 имеет вид

Y0 =

21.0000	-32.4853	21.0000	-15.5147	7.0000
11.1569	-15.7782	9.2426	-6.5355	2.9142
1.0000	0.0000	0.0000	0.0000	0.0000
-0.1569	0.5355	0.7574	-0.2218	0.0858
0.0000	0.0000	1.0000	0.0000	0.0000

Считая данное решение точным, оценим нормы невязок для предыдущих решений.

norm(Y0 - Y)	norm(Y0 - Y1)	norm(Y0 - Y2)	norm(Y0 - Y3)
1.2635e-013	1.2635e-013	6.3827e-014	0.0065

Из анализа следует, что функции expm1(A) и expm2(A) дают результаты с погрешностью в пределах ошибки округления, а функция expm3(A) имеет ту же погрешность.

Сопутствующие функции: EXP, FUNM, LOGM, SQRTM.

Ссылки:

1. Golub G. H., Van Loan. Matrix Computation. Oxford: John Hopkins University Press, 1983.
2. Moler C. B., Van Loan. Nineteen Dubious Ways to Compute the Exponential of a Matrix// SIAM Review. 1979. Vol. 20. P. 801-836.
3. Higham N. J. The Test Matrix Toolbox for MATLAB (version 3.0)//*Numerical Analysis Report.* Vol. 276. Manchester, 1995.
4. Потемкин В. Г. *Пакет программ JORD.* М.: МИФИ, 1995.

LOGM — Вычисление логарифма матрицы

Синтаксис:

```
Y = logm(A)
[Y, esterr] = logm(A)
```

Описание:

Функция Y = logm(A) вычисляет функцию log(A), такую, что для большинства матриц A должно выполняться условие

logm(expm(A)) = A = expm(logm(A)).

При таком обращении возможно появление диагностического предупреждения

Warning: LOGM appears inaccurate. esterr = xxx

Предупреждение: Функция LOGM вычислена неточно. esterr = xxx

Если матрица A - действительная симметрическая или комплексная эрмитова, то теми же свойствами обладает и функция logm(A).

Функция [Y, esterr] = logm(A) кроме вычисленной матрицы возвращает оценку погрешности в виде относительной невязки

norm(expm(Y) - A) / norm(A).

В этом случае диагностическое сообщение не выводится.

Замечание:

Функцию матричной экспоненты logm(A) не следует путать с функцией log(A), которая вычисляет логарифм от каждого элемента массива A.

Пример:

Рассмотрим матрицу A = expm(chebspec(5)), вычисленную в предыдущем разделе.

```
A =
   21.0000-32.4853 21.0000-15.5147  7.0000
   11.1569-15.7782  9.2426 -6.5355  2.9142
    1.0000  0.0000  0.0000  0.0000  0.0000
   -0.1569  0.5355  0.7574 -0.2218  0.0858
    0.0000  0.0000  1.0000  0.0000  0.0000
```

logm(A)

Warning: LOGM appears inaccurate. esterr = 6.471e-007

logm(A)				
5.5000-0.0000i	-6.8284+0.0000i	2.0000-0.0000i	-1.1716+0.0000i	0.5000-0.0000i
1.7071-0.0000i	-0.7071+0.0000i	-1.4142-0.0000i	0.7071+0.0000i	-0.2929-0.0000i
-0.5000-0.0000i	1.4142+0.0000i	0.0000-0.0000i	-1.4142+0.0000i	0.5000-0.0000i
0.2929-0.0000i	-0.7071+0.0000i	1.4142-0.0000i	0.7071+0.0000i	-1.7071-0.0000i
-0.5000-0.0000i	1.1716+0.0000i	-2.0000-0.0000i	6.8284+0.0000i	-5.5000-0.0000i

Алгоритм:

Функция logm, как и другие функции от матриц, вычисляется с использованием алгоритма Парлетта [1]. Этот алгоритм использует приведение к форме Шура и может давать неточные или полностью несостоятельные результаты в случае кратных собственных значений.

Сопутствующие функции: EXPM, FUNM, SQRTM.

Ссылки:

1. Golub G. H., Van Loan. Matrix Computation. Oxford: John Hopkins University Press, 1983.

SQRTM **Вычисление функции $A^{1/2}$**

Синтаксис:

Y = sqrtm(A)

[Y, esterr] = sqrtm(A)

Описание:

Функция Y = sqrtm(A) вычисляет одну из многих матриц, которые удовлетворяют условию Y * Y = A.

При таком обращении возможно появление диагностического предупреждения

Warning: SQRTM appears inaccurate. esterr = xxx

Предупреждение: Функция SQRTM вычислена неточно. esterr = xxx

Если матрица A - действительная симметрическая или комплексная эрмитова, то теми же свойствами обладает и функция sqrtm(A).

Функция [Y, esterr] = sqrtm(A) кроме вычисленной матрицы возвращает оценку погрешности в виде относительной невязки

norm(Y * Y - A) / norm(A).

В этом случае диагностическое сообщение не выводится.

Замечание:

Функцию sqrtm(A) не следует путать с функцией sqrt(A), которая вычисляет положительный квадратный корень от каждого элемента массива.

Примеры:

Рассмотрим матричное представление разностного оператора 4-го порядка.

```
A =
     5    -4     1     0     0
    -4     6    -4     1     0
     1    -4     6    -4     1
     0     1    -4     6    -4
     0     0     1    -4     5
```

Эта матрица симметрическая и положительно определенная; ее единственный положительно определенный квадратный корень представляет собой разностный оператор 2-го порядка:

```
Y = sqrtm(A)
Y =
    2.0000   -1.0000    0.0000    0.0000    0
   -1.0000    2.0000   -1.0000    0.0000    0.0000
    0.0000   -1.0000    2.0000   -1.0000    0.0000
    0.0000    0.0000   -1.0000    2.0000   -1.0000
    0.0000    0.0000    0.0000   -1.0000    2.0000
```

Матрица вида

```
X =
     7    10
    15    22
```

имеет 4 матрицы, являющиеся ее квадратным корнем.
Две из них следующие:

Y1 = 1.5667 1.7408 2.6112 4.1779	Y2 = 1 2 3 4

Две другие - соответственно -Y1 и -Y2.

Все 4 матрицы могут быть получены на основе спектрального разложения исходной матрицы

[R, D] = eig(X).

R = -0.8246 -0.4160 0.5658 -0.9094	D = 0.1386 0 0 28.8614

S1 = sqrt(D) 0.3723 0 0 5.3723	S2 = -0.3723 0 0 5.3723

Y1 = R * S1 / R 1.5667 1.7408 2.6112 4.1779	Y2 = R * S2 / R 1.0000 2.0000 3.0000 4.0000

Функция sqrtm строит решение только для положительных значений квадратных корней, и результатом является матрица Y1, хотя матрица Y2 представляется более предпочтительным решением, поскольку она целочисленна.

Алгоритм:

Функция sqrtm является просто аббревиатурой для вызова функции funm(A, 'sqrt'). Алгоритм, реализующий функцию funm, использует приведение к форме Шура и может давать неточные или полностью несостоятельные результаты в случае кратных собственных значений.

Сопутствующие функции: EXPM, FUNM, LOGM.

FUNM **Вычисление произвольных функций от матрицы**

Синтаксис:

Y = funm(A, '<*имя функции*>')

[Y, esterr] = funm(A, '<*имя функции*>')

Описание:

Функция Y = funm(A, '<*имя функции*>') позволяет вычислить любую функцию от матрицы, если она имеет имя, составленное из латинских букв. Это могут быть, например, все элементарные математические функции.

При таком обращении возможно появление диагностического предупреждения

Warning: Result from FUNM may be inaccurate. esterr = xxx

Предупреждение: Результат вычисления функции FUNM может быть неточным. esterr = xxx

Функция [Y, esterr] = logm(A) кроме вычисленной матрицы возвращает оценку погрешности в виде относительной невязки norm(expm(Y) - A) / norm(A).

В этом случае диагностическое сообщение не выводится.

Функции funm(A, 'sqrt') и funm(A, 'log') эквивалентны функциям sqrtm(A) и logm(A). Функции funm(A, 'exp') и expm(A) вычисляют одну и ту же функцию, но различными способами; применение функции expm предпочтительнее.

Пример:

Следующие последовательности операторов в пределах ошибок округления должны давать одинаковые результаты.

```
S = funm(A, 'sin');          E = expm(i * A);
C = funm(A, 'cos');          C = real(E);
                             S = imag(E);
```

В любом случае они удовлетворяют условию S*S + C*C = I, где I = eye(size(A)).

Алгоритм:

Функция funm вычисляет функции от матриц с использованием алгоритма Парлетта, описанного в работах [1,2]. Этот алгоритм потенциально неустойчив. Если матрица имеет кратные или близкие к ним собственные значения, то функция funm дает неточные или полностью несостоятельные результаты. При разработке алгоритма была сделана попытка выявить эту ситуацию и дать диагностическое сообщение. Однако выбранный критерий столь чувствителен, что сообщение может быть выдано даже при точном результате.

Если матрица A действительная симметрическая или комплексная эрмитова, то ее форма Шура диагональна и результаты могут обладать очень высокой точностью.

Сопутствующие функции: EXPM, SQRTM, LOGM.

Ссылки:

1. Golub G. H., Van Loan. *Matrix Computation*. Oxford: John Hopkins University Press, 1983.
2. Moler C. B., Van Loan. Nineteen Dubious Ways to Compute the Exponential of a Matrix//*SIAM Review*. 1979. Vol. 20. P. 801-836.

Полиномы и операции над ними

POLYVAL **Вычисление полинома**

Синтаксис:

```
Y= polyval(p, X)
[Y, DELTA] = polyval(P, X, S)
```

Описание:

Функция Y= polyval(p, X), где $p = [p_1\ p_2\ \dots\ p_n\ p_{n+1}]$ - вектор коэффициентов полинома $p(x) = p_1x^n + p_2x^{n-1} + \dots + p_nx + p_{n+1}$, вычисляет значение этого полинома в точках, заданных массивом X.

Функция [Y, DELTA] = polyval(p, X, S), где S - вспомогательный массив записей, сформированный функцией polyfit и предназначенный для оценки ошибок вычисления значений полинома. Если ошибки в данных для модуля polyfit независимы и подчиняются норманову закону распределения с постоянной дисперсией, то доверительный интервал для значений массива Y ± DELTA соответствует 50 %.

Пример:

Вычислим значение полинома $p(x) = 3x^2 + 2x + 1$ в точке x = 5.

```
p = [3  2  1]
p =    3    2    1
y = polyval(p, 5)
y =   86
```

Сопутствующие функции: POLYDER, POLYVALM.

POLYVALM **Вычисление матричного полинома**

Синтаксис:

```
Y = polyval(p, S)
```

Описание:

Функция Y = polyval(p, S), где $p = [p_1\ p_2\ \dots\ p_n\ p_{n+1}]$ - вектор коэффициентов матричного полинома

$$p(X) = p_1X^n + p_2X^{n-1} + \dots + p_nX + p_{n+1},$$

вычисляет значение этого полинома для матрицы X = S.

Пример:

Рассмотрим матрицу Паскаля порядка **4**, составленную из биномиальных коэффициентов.

```
S = pascal(4)
S =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
```

Вычислим характеристический полином этой матрицы.

```
p = poly(S)
p = 1.0000  -29.0000   72.0000  -29.0000    1.0000 3    2    1
```

Сравним результаты применения функций polyvalm и polyval.

polyvalm(p, S)				polyval(p, S)			
1.0e-010 *							
-0.0027	-0.0094	-0.0222	-0.0428	16.00	16.00	16.00	16.00
-0.0094	-0.0326	-0.0749	-0.1423	16.00	15.00	-140.00	-563.00
-0.0222	-0.0749	-0.1713	-0.3233	16.00	-140.00	-2549.00	-12089.00
-0.0428	-0.1423	-0.3233	-0.6091	16.00	-563.00	-12089.00	-43779.00

Матрица polyvalm(p, S) в пределах погрешности - нулевая матрица, что подтверждает теорему Кэли - Гамильтона о том, что всякая матрица удовлетворяет своему характеристическому уравнению.

Матрица polyval(p, S) - это массив значений полинома p для каждого элемента матрицы S.

Сопутствующие функции: POLYDER, POLYVAL.

CONV **Умножение полиномов**

Синтаксис:

```
c = conv(a, b)
```

Описание:

Если заданы полиномы a и b длины степеней соответственно m и n, то их произведение - это полином c степени m + n, k-й элемент которого находится по формуле

$$c(k) = \sum_{j=\max(1,k+1-n)}^{\min(k,m)} a(j)b(k+1-j).$$

Функция z = conv(x, y) вычисляет произведение двух полиномов a и b.

Пример:

Найдем произведение полиномов $x^3 + 2x^2 + 3x + 4$ и $10x^2 + 20x + 30$. Для этого сформируем векторы a = [1 2 3 4] и b = [10 20 30] и вычислим

```
c = conv(a, b)
c = 10   40   100   160   170   120
```

Сопутствующие функции: DECONV, RESIDUE.

DECONV **Деление полиномов**

Синтаксис:

```
[q, r] = deconv(c, a)
```

Описание:

Функция [q, r] = deconv(c, a) реализует деление полинома c на полином a; частное от деления возвращается в виде вектора q, остаток - в виде вектора r, так что выполняется соотношение c = conv(q, a) + r.

Пример:

Если c = [10 40 100 160 170 120] и a = [1 2 3 4], то деление этих полиномов дает следующий результат:

```
[q, r] = deconv(c, a)
q = 10   20   30
r = 0   0   0   0   0   0,
```

и этот результат соответствует примеру из раздела CONV.

Сопутствующие функции: CONV.

POLYDER **Вычисление производных**

Синтаксис:

```
dp = polyder(p)
dp = polyder(a, b)
[q, p] = polyder(b, a)
```

Описание:

Функция dp = polyder(p) возвращает производную полинома dp(x)/dx.

Функция dp = polyder(a, b) возвращает производную от произведения полиномов a(x) * b(x).

Функция [q, p] = polyder(b, a) возвращает производную от отношения полиномов b(x)/a(x) в виде отношения полиномов q(x)/p(x).

Примеры:

Вычислим производную полинома $p(x) = 3x^2 + 2x + 1$.

```
p = [3 2 1];  dp = polyder(p)
dp =    6    2
```

Вычислим производную произведения полиномов.

a	b	dp = polyder(a, b)
1 3 5 7	3 2 1	15 44 66 68 19

Вычислим производную отношения двух полиномов b(x)/a(x).

b	a	[q, p] = polyder(b, a)
3 2 1	1 3 5 7	$\frac{q}{p} = \frac{[-3 \quad -4 \quad 6 \quad 36 \quad 9]}{[1 \quad 6 \quad 19 \quad 44 \quad 67 \quad 70 \quad 49]}$

Сопутствующие функции: POLYFIT, POLYVALM.

ROOTS **Вычисление корней полинома**

Синтаксис:

r = roots(p)

Описание:

Функция r = roots(p), где $p = [p_1\ p_2 \ldots p_n\ p_{n+1}]$ - ***вектор-строка*** коэффициентов полинома $p(x) = p_1x^n + p_2x^{n-1} + \ldots + p_nx + p_{n+1}$, вычисляет ***вектор-столбец*** корней этого полинома.

Функция p = poly(r), где r - ***вектор-столбец*** корней некоторого полинома, вычисляет ***вектор-строку*** коэффициентов этого полинома.

Пример:

Вычислим корни полинома $p(x) = x^3 + 3x^2 + 5x + 7$.

p	r = roots(p)
1 3 5 7	-2.1795 -0.4102 + 1.7445i -0.4102 - 1.7445i

p = poly(r)

p = 1.0000 3.0000 5.0000 7.0000 + 0.0000i

Сопутствующие функции: POLY.

POLY **Вычисление характеристического полинома**

Синтаксис:

p = poly(A)

p = poly(r)

Описание:

Функция p = poly(A), где A - матрица порядка n, вычисляет ***вектор-строку*** коэффициентов характеристического полинома $p(s) = \det(sI - A) = p_1s^n + p_2s^{n-1} + \ldots + p_ns + p_{n+1}$.

Функция p = poly(r), где r - ***вектор-столбец*** корней некоторого полинома, вычисляет ***вектор-строку*** коэффициентов этого полинома.

Пример:

Рассмотрим рациональную матрицу A, вычисляя коэффициенты характеристического полинома, его корни и по ним вновь восстановим характеристический полином:

```
A =
     -5/3      -1        -2/3
      -5/6      1/4      11/12
       1/6     -17/4    -19/12
p = poly(A)
p =    1         3         5         7
r = roots(p)
r =    -2.1795
  -0.4102 + 1.7445i
  -0.4102 - 1.7445i
p = poly(r)
p =    1.0000          3.0000          5.0000          7.0000 + 0.0000i
```

Сопутствующие функции: ROOTS, RESIDUE.

RESIDUE, RESI2 **Разложение на простые дроби**

Синтаксис:

```
[r, p, k] = residue(b, a)
coeff = resi2(u, v, pole, n, k)
[b, a] = residue(r, p, k)
```

Описание:

Функция p = [r, p, k] = residue(b, a) вычисляет вычеты, полюса и многочлен целой части отношения двух полиномов b(s) и a(s):

простые корни:

$$\frac{b(s)}{a(s)} = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \ldots + \frac{r_n}{s-p_n} + k(s);$$

- входные переменные - векторы b и a определяют коэффициенты полиномов числителя и знаменателя по убывающим степеням s;
- выходные переменные - *вектор-столбец* r вычетов, *вектор-столбец* p полюсов и *вектор-строка* k целой части дробно-рациональной функции;
- количество полюсов определяется по формуле
  n = length(a) - 1 = length(r) = length(p);
- вектор коэффициентов многочлена прямой передачи будет пустым, если length(b) < length(a); в противном случае length(k) = length(b) -length(a) + 1;

кратные корни:

если p(j) = . . . =p(j + m - 1) - полюс кратности m, то разложение на простые дроби включает следующий член [1]:

$$\frac{r_j}{s-p_j}+\frac{r_{j+1}}{(s-p_j)^2}+\ldots+\frac{r_{j+m-1}}{(s-p_j)^m}.$$

Функция rj = resi2(b, a, pole, m, j) вычисляет вектор коэффициентов разложения дробно-рациональной функции b(s)/a(s) для полюса pole, имеющего кратность m. Параметр j указывает, какой из коэффициентов r_j вычисляется при данном обращении к функции; по умолчанию j = m; если не указано m, то оно принимается за 1, то есть функция определяет вычеты для простых корней.

Функция [b, a] = residue(r, p, k) с тремя входными и двумя выходными параметрами выполняет обратную функцию свертки разложения в дробно-рациональную функцию отношения двух полиномов b(s) и a(s).

Пример 1:

Рассмотрим дробно-рациональную функцию отношения двух полиномов b(x)/a(x) с некратными корням.

$$\frac{b}{a}=\frac{[3\quad 2\quad 1]}{[1\quad 3\quad 5\quad 7]};$$

```
[r, p, k] = residue(b, a)
r =                          p =                          k =
  1.7642                       -2.1795                      [ ]
  0.6179 + 0.7589i             -0.4102 + 1.7445i
  0.6179 - 0.7589i             -0.4102 - 1.7445i
```

Поскольку порядок числителя меньше порядка знаменателя, целая часть функции отсутствует.

Пример 2:

Поменяем местами числитель и знаменатель дробно-рациональной функции.

$$\frac{a}{b}=\frac{[1\quad 3\quad 5\quad 7]}{[3\quad 2\quad 1]};$$

```
[ra, pa, ka] = residue(a, b)
ra =                         pa =                         ka =
  0.5185 - 1.8332i             -0.3333 + 0.4714i            0.3333   0.7778
  0.5185 + 1.8332i
```

В этом случае появляется целая часть функции, определяемая вектором ka.

Пример 3:

Обратимся к случаю кратных корней и рассмотрим следующую дробно-рациональную функцию:

$$\frac{b}{a}=\frac{[3\quad 2\quad 1]}{[1\quad 3\quad 3\quad 1]};$$

```
r1 = resi2(b, a1, -1, 3, 1)      r2 = resi2(b, a1, -1, 3, 2)      r3 = resi2(b, a, -1, 3)
          3                                -4                               2
```

Пример 4:

Обратимся к результатам примера 1. Зная их, восстановим исходную дробно-рациональную функцию, используя следующее обращение:

```
[b1, a1] = residue(r, p, k)
```

$$\frac{b1}{a1} = \frac{[3 \quad 2 \quad 1]}{[1.0000 \quad 3.0000 \quad 3.0000 \quad 7.0000 + 0.0000i]}$$

```
norm(a - a1) = 4.3739e - 015
```

В пределах погрешности компьютера результаты совпадают.

Ограничения:

В вычислительном плане разложение дробно-рациональной функции на простые дроби плохо обусловлено. Если полином знаменателя имеет корни, близкие к кратным, то малые возмущения исходных данных могут привести к большим погрешностям вычисления полюсов и вычетов. Предпочтительнее использовать описание в пространстве состояний или представление таких функций в виде нулей и полюсов.

Сопутствующие функции: POLY, ROOTS.

Ссылки:

1. Oppenheim A. V., Schafer R. W. *Digital Signal Processing*. Englewood Cliffs, N. Y.: Prentice-Hall, 1975, P. 56-58.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ