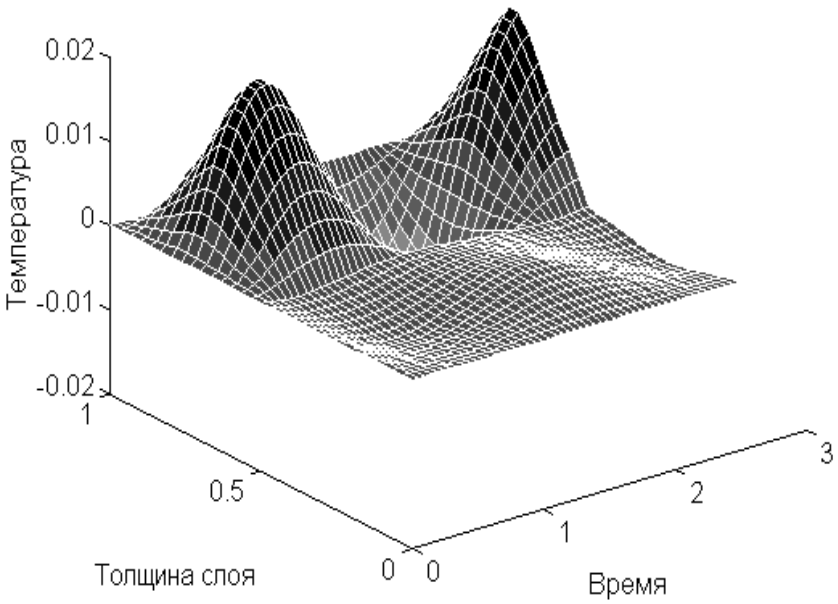
В.И.Паасонен

Инструмент научных исследований

M A T L A B



МИНИСТЕРСТВО ОБЩЕГО
И ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

В.И. Паасонен

Инструмент научных исследований

MATLAB

(УЧЕБНОЕ ПОСОБИЕ)

НОВОСИБИРСК
2000

ББК 32.973
УДК 681.3.06

Паасонен В.И. Инструмент научных исследований MATLAB. Учеб. пособие. Новосиб. ун-т. Новосибирск, 2000. - 61 с.

Настоящее учебное пособие представляет собой введение в MATLAB и имеет целью на отдельных фрагментах вычислительных задач познакомить читателя с этой многофункциональной системой, широко распространенной во всем мире и с недавнего времени используемой также в некоторых университетах России. Пособие написано для сопровождения специального курса лекционных и практических занятий, но может быть использовано также для самостоятельного изучения системы. Оно познакомит читателя с приемами работы и возможностями системы Matlab и позволит приобрести достаточные навыки для ее применения в вычислительных задачах.

Пособие предназначено для аспирантов и студентов преимущественно негуманитарных специальностей, а также для научных работников, желающих включить в круг своих инструментальных средств эту универсальную систему.

Р е ц е н з е н т
доктор физ.-мат. наук, доцент Л. Б. Чубаров

Рекомендовано редакционно-издательским советом НГУ для специальностей 0647, 2013, 2014

ПРЕДИСЛОВИЕ

Созданию первой версии системы Matlab (сокращение от Matrix Laboratory) в начале восьмидесятых годов предшествовало длительное и бурное развитие вычислительных методов. С расширением круга решаемых задач в областях сложной геометрии, с развитием метода расщепления многомерных задач и метода конечных элементов численные методы линейной алгебры получили свое второе рождение.

Потребности практики, многообразие методов и расширяющиеся возможности вычислительной техники создали необходимые и достаточные предпосылки для создания компьютерной технологии решения математических задач, тесно связанных с линейной алгеброй. Результатом реализации этой идеи явилась система Matlab, разработанная фирмой The MathWorks. Ins. (США, Массачусетс). Одновременно создавались другие специализированные и универсальные системы, в частности, Mathcad, Maple и Mathematica. Система Mathcad удобна для оформления материалов, содержащих множество формул и результатов вычислений, Maple ориентирована на выполнение символьных вычислений. Mathematica пытается охватить все и поражает своей универсальностью, но уступает упомянутым системам в аспектах, соответствующих их специализации. Конкуренция побуждает разработчиков создавать аналоги наиболее привлекательных элементов, имеющихся в других системах, или решать проблему непосредственного доступа к ним. Эта тенденция размывает некогда ярко выраженную специализацию различных пакетов. Не исключено, что в недалеком будущем этот процесс приведет к поглощению некоторых систем или к их слиянию.

Система Matlab от версии к версии пополнялась и модифицировалась в своем специализированном направлении – линейной алгебре и работе с массивами, и параллельно оснащалась элементами, расширяющими ее возможности и улучшающими интерфейс. Сегодня она представляет собой развитую интерактивную систему для выполнения расчетов в различных областях научной и инженерно-технической деятельности.

Поддерживая многочисленные операции с векторами, матрицами и полиномами, Matlab также решает задачи аппроксимации, интерполяции и оптимизации, решает нелинейные алгебраические, дифференциальные и разностные уравнения, вычисляет квадратуры и выполняет преобразования Фурье, обеспечивает графическое изображение функций в разнообразных формах и цветовых решениях. Пользователю предоставлена возможность работы с комплексными числами, специальными функциями и пакетом статистической обработки информации. Кроме того, Matlab располагает управляющими элементами, позволяющими создавать приложения с графическим интерфейсом пользователя так же, как в Visual Basic или Delphi. Наконец, Matlab имеет собственный входной язык с привлекательными чертами в реализации матричных операций, обеспечивающий гибкость, многоплановость работы программ в зависимости от числа и типа входных и возвращаемых параметров.

Недостатком системы является присущая всем интерпретаторам медлительность, но она в значительной мере компенсируется экономией времени на создание программ, так как в библиотеках Matlab имеются готовые функции для расчета многих элементарных вычислительных задач. Впрочем, если отдельные блоки программы требуют значительного объема вычислений, их можно писать на C или Фортране – смешанное программирование Matlab также поддерживает.

Настоящее учебное пособие не является полным описанием системы Matlab. Цель состоит в том, чтобы дать общее представление о системе, познакомить читателя с конструкциями языка и основными приемами работы в Matlab, на отдельных примерах продемонстрировать многообразие его функциональных возможностей и побудить читателя использовать эту добротно сделанную систему.

За основу курса намеренно принята довольно простая версия Matlab 4.2, так как она не отличается кардинально от самой послед-

ней версии Matlab 5.3, но менее требовательна к ресурсам и, что еще важнее, более удобна для изучения. Полезность изучения именно этой версии связана с тем, что она не перегружена сервисом и поэтому позволяет глубже понять программно–технологическую суть выполняемых операций.

Автор выражает признательность коллегам по кафедре математического моделирования и по математическому факультету НГУ, поддержавшим выпуск данного пособия, и особо благодарит Л. Б. Чубарова, высказавшего в качестве рецензента ценные замечания, учет которых способствовал совершенствованию структуры пособия в целом и улучшению содержания отдельных его фрагментов. Замечания и предложения по содержанию и развитию курса автор примет с благодарностью.

В.И. Паасонен
Апрель-сентябрь 2000 года
Телефон: (3832) 34-36-56
Эл. адрес: paas@ict.nsc.ru

ИВТ СО РАН
пр. Академика Лаврентьева, 6
Новосибирск, 630090, Россия

ОПЕРАЦИОННАЯ СРЕДА MATLAB

Настройка системы

Запуск системы Matlab 4.2c осуществляется двойным щелчком по его пиктограмме и вызывает появление командного окна с приглашением на ввод команды в виде знака «>>». Первому приглашению в сеансе работы предшествует подсказка о существовании нескольких ознакомительных команд: intro (краткое введение), demo (интерактивная демонстрационная система), help help (справка о пользовании справочной системой) и т.п.

Каждой команде (или функции) Matlab, за исключением встроенных команд, соответствует одноименный файл с расширением m (m–файл). Так при вызове функции diff(x) в действительности исполняется программа на языке Matlab, записанная в файле diff.m, или, кратко говоря, исполняется файл diff.m. При исполнении команд соответствующие m–файлы отыскиваются системой в «стандартных» местах, указанных в переменной окружения path. Она является внутренней переменной системы, действует на протяжении сеанса Matlab, и ее изменение никак не отражается на одноименной переменной окружения path операционной системы.

Для организации доступа к личным или иным подключаемым библиотекам m–функций необходимо изменить переменную path, включив в нее каталоги, содержащие соответствующие m–файлы. Так команда path(path,'c:\work')пополняет текущий перечень путей доступа каталогом c:\work. После такого подключения пользовательские m–файлы из каталога c:\work будут исполняться так же, как системные. Заметим, что программы могут быть исполнены также по команде меню File – Run m–file... Команда path без параметров выдает текущий перечень установленных путей, – таким способом можно контролировать правильность подключения.

Каждая команда обрабатывается немедленно, при этом значения выходных параметров выводятся в командное окно, а для рисунков открываются специальные графические окна. Формат вывода чисел выбирается с помощью команды меню Options -Numeric Format. Атрибуты командного окна (шрифт, цвет и пр.) пользователь может установить по своему вкусу в диалоговом окне, открывающемся по команде меню Options – Command Window Font.

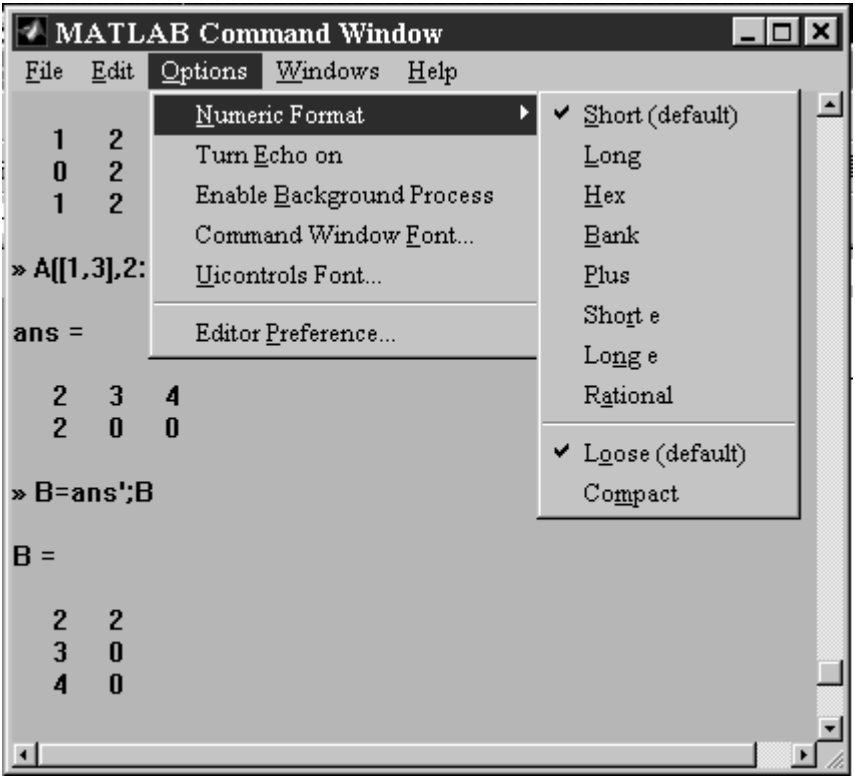


Рис.1. Командное окно.

Настройка на внешний текстовый редактор[1] осуществляется в окне, появляющемся по команде меню Options –Editor Preference. Выбранный редактор будет всякий раз открываться для подготовки m–файлов по командам меню Open M–file или New – M–file.

При запуске Matlab первым автоматически исполняется файл matlabrc.m, из которого, в свою очередь, вызывается файл startup.m, если он существует. В первом устанавливаются стандартные параметры среды Matlab (т.е. принятые по умолчанию), во втором пользователь имеет возможность подправить их по своему усмотрению, в частности, установить пути доступа, физические константы, размеры графического окна и другие параметры, формирующие инди-

[1] Версии Matlab 5.x имеют собственный текстовый редактор

видуальную рабочую среду, отличную от стандартной. При инсталляции система не содержит файла startup.m, и если в нем есть необходимость, его надо создать и записать в корневой каталог системы Matlab. При сетевой установке Matlab файл matlabrc.m обычно находится в ведении администратора системы, формирующего общую для всех стандартную среду, а каждый пользователь имеет возможность сформировать индивидуальный файл startup.m с коррективами.

Все команды, исполняемые в течение сеанса, запоминаются системой. Они загружаются в «бесконечный» стек, и до завершения сеанса их можно оттуда извлекать. Поэтому иногда нет необходимости набирать команду на клавиатуре, а достаточно найти в предыстории сеанса такую же или похожую команду, отредактировать ее и исполнить нажатием <Enter>. Клавиша <↑> вызывает появление последней команды, повторное нажатие переводит к предпоследней, и т.д. Стек команд закольцован, и в обратной последовательности команды вызываются клавишей <↓> (первая команда сеанса, вторая и т.д.).

Для переходов вверх и вниз по командному окну на одну строку следует использовать сочетания клавиш <Ctrl+P>, <Ctrl+W>, так как клавиши <↑> и <↓> заняты иными функциями. Это отступление от универсальности, принятой в приложениях Windows, может несколько раздражать пользователя. Впрочем, такие переходы в командном окне можно осуществить иначе – щелчком мыши по элементам линейки прокрутки или клавишами <Page Up> и <Page Down>.

Сеанс работы в системе Matlab завершают командами quit или exit из командного окна, командой меню File – Exit Matlab, или, наконец, путем закрытия окон Matlab средствами Windows.

Некоторые элементы языка

В системе Matlab константы записываются как в других языках, синтаксические ограничения на имена переменных традиционны и даже несколько старомодны. Как в DOS ограничивается длина, запрещаются пробелы и специальные символы, кроме подчерка. Однако типы и размерности переменных вовсе не описываются: они определяются автоматически текущим содержанием перемен-

ных, а система при этом располагает функциями для запросов о них.

Арифметические операции сложения, вычитания, умножения, левого и правого деления, возведения в степень и операция матричного сопряжения обозначаются в языке Matlab символами +, -, *, \, /, ^ и ' соответственно. В системе реализовано два вида операций. *Операции над матрицами (векторами) полностью соответствуют правилам линейной алгебры, а операции над массивами выполняются поэлементно.* В случаях сложения и вычитания эти виды операций очевидно совпадают.

При любой несогласованности размерностей выдается сообщение об ошибке.

Для поэлементных операций в качестве знака отличия от матричных используется префикс к знаку операции в виде точки[1]. Например, запись a.^b при совпадающей размерности операндов означает возведение элементов массива a в степень элементов массива b поэлементно. Операции над массивами допускают также вырождение операндов a и b в скаляры. Так, в нашем примере в одном случае скаляр a будет возводиться в различные степени, определяемые массивом b, а в другом все элементы массива a будут возводиться в одну и ту же степень b.

Ниже в таблице 1 дано описание символов, используемых в языке Matlab.

Таблица 1

РАЗДЕЛИТЕЛИ И СПЕЦИАЛЬНЫЕ СИМВОЛЫ MATLAB

Символы	Описание их назначения в языке Matlab
[ ]	Заключают список при формировании массива. Заключают возвращаемые параметры функции, если их число превышает единицу.
( )	Указывают порядок выполнения операций. Заключают список индексов массива. Заключают список входных параметров функции.
=	Знак присваивания выражения переменной.

[1] Точка с апострофом (.') означает простое транспонирование без операции сопряжения.

.	Десятичная точка в константах. Знак-префикс поэлементной операции.
..	Переход вверх на один уровень по дереву каталогов.
...	Знак продолжения текущего оператора на следующей строке. Более чем три точки интерпретируются так же.
,	Разделяет индексы массива в списке. Разделяет элементы списка при формировании строки массива (то же делает и пробел). Разделяет операторы без подавления вывода. Разделяет входные и выходные параметры функции.
;	Разделяет строки при формировании массивов. Разделяет операторы с подавлением вывода.
:	Указывает диапазон в списке при формировании массива. Указывает столбец или строку массива.
пробел	Разделяет элементы списка при формировании строки массива (то же делает и запятая).
%	Указатель строки комментария или логического конца строки (текст за символом % игнорируется).
'	Транспонирование с комплексным сопряжением.

Логические операции 'и', 'или' и 'не' обозначаются символами «&», «|» и «~». Для исключительного 'или' не предусмотрено отдельного символа; эта операция реализована как функция xor(A,B). Для формирования операций отношения используются комбинации символов сравнения «<», «>», «=» и знак отрицания «~». Для логических операций и для операций отношения в языке Matlab применяется следующий синтаксис:

```
A & B        A и B
A | B        A или B
~B           не B

A < B        A меньше B
A > B        A больше B
A < = B      A меньше или равно B
```

A > = B	A больше или равно B
A = = B	A равно B
A ~ = B	A не равно B

Пробелы в записи данных операций (и многих других, за исключением операций над строками) имеют только эстетическое значение, а на результате не отражаются. Так выражения A&~B, A &~B и A & ~B эквивалентны.

Любое число в системе Matlab рассматривается одновременно и как булево значение. Число нуль при этом соответствует значению false, а число, отличное от нуля – значению true. С другой стороны, булевы значения false и true система отождествляет с числами 0 и 1 соответственно. *Эта особенность Matlab требует осторожности при формировании логических выражений* – нужно, например, избегать записи системы неравенств a<x & x<b в сокращенной форме a<x<b, поскольку последняя интерпретируется как (a<x)<b и формально не запрещена, а по существу ошибочна. Так, например, Matlab оценивает неравенство 0<x<2 как истинное при всех x. В самом деле, первое неравенство 0<x оценивается числом 0 или 1 в зависимости от того, ложно оно или истинно, а затем этот результат сравнивается с 2, а он в обоих случаях меньше 2 (0<2 и 1<2).

В отличие от других языков Matlab располагает только двумя видами циклов

```
for k = <выражение-массив>
<операторы>
end,

while <логическое выражение>
<операторы>
end,
```

однако первый из них более содержателен, чем его аналоги в универсальных языках программирования, так как здесь переменная цикла k не обязательно является скалярной величиной, она может принимать значения столбцов из выражения-массива и, следовательно, операторы в теле цикла могут выполнять векторные операции. Однако на практике k чаще всего принимает скалярные значе-

ния из последовательности n:d:m (от n до m с шагом d), как и в других языках. Выполнение цикла может быть прервано оператором break в сочетании с условным оператором if.

Условные выражения в языке Matlab аналогичны применяемым в других языках, но в нем не используется слово then:

```
if <логическое выражение>
    <операторы>
elseif <логическое выражение>
    <операторы>
else <логическое выражение>
    <операторы>
end
```

Блоков типа elseif в пределах условного выражения может быть сколько угодно. Блок else, а также и блоки elseif , могут отсутствовать вовсе. *Следует отличать оператор* elseif *от пары операторов* else if. Следующие два фрагмента, реализующие один и тот же алгоритм формирования матрицы с числами -1, 0 и 1 под, на и над главной диагональю соответственно, иллюстрируют это различие.

```
if i == j          % на диагонали
  a(i,j) = 0;
elseif i > j       % под диагональю
  a(i,j) = -1;
else               % над диагональю
  a(i,j) = 1;
end                % конец первого фрагмента

if  i = = j        % на диагонали
  a(i,j) = 0;
else               % не на диагонали:
  if i > j         % под диагональю
    a(i,j) = -1;
  else             % над диагональю
    a(i,j) = 1;
  end              % конец вложенного if-блока
end                % конец второго фрагмента
```

В первом фрагменте elseif начинает очередную ветвь условного выражения, во втором – оператор if, следующий за else, начинает

самостоятельное (вложенное) условное выражение, которое естественно заканчивается отдельным оператором end.

Команды и рабочая область

Познакомившись в предыдущем разделе с описанием элементов языка Matlab, сформируем несколько переменных. Выполним из командного окна команду[1] формирования матрицы

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

Завершение командной строки символом «;» означало бы подавление вывода результата в командное окно. Наша команда не заканчивается точкой с запятой, поэтому в командном окне появляется отклик системы в виде

```
A =
   1   0   0   4
   0   2   1   0
   1   2   0   0
```

Размерность любой переменной возвращает функция size. В данном случае size(A) = 3 4, т.е. размерность представляет собой вектор-строку, состоящую из двух натуральных чисел –числа строк и числа столбцов переменной A. Команда

```
>>A([1,3],2:4)
```

формирует новую матрицу, составленную из элементов матрицы A, находящихся на пересечении строк с номерами 1 и 3 и столбцов с номерами 2,...,4, присваивая результат служебной переменной ans (answer), которая всегда содержит результат вычисления последнего выражения, если он не был присвоен какой-нибудь переменной. В данном случае

```
ans =
    0   0   4
    2   0   0
```

[1] При формировании матрицы строки разделяются точкой с запятой, а элементы строки – запятой или пробелом

С переменной ans можно оперировать как с любой другой. Ниже матрица ans транспонируется, и результат присваивается переменной B, при этом вывод результата подавляется. Затем командой, состоящей исключительно из идентификатора переменной B, запрашивается ее содержимое.

```
>>B=ans'; B

    B=
     0   2
     0   0
     4   0
```

Мы видим, что в одной строке допускается несколько команд. Они должны быть разделены символами «,» или «;». При этом разделитель «,» не препятствует выводу результата в командное окно, если он предусмотрен командой, а разделитель «;» подавляет вывод.

Для матриц с большим количеством нулей полезен перевод в специальное представление (разреженной структуры), при котором хранятся только ненулевые элементы и их индексы. Функция sparse осуществляет перевод полной матрицы в разреженную:

```
>>S=sparse(B)
   S =

     (3,1)   4
     (1,2)   2
```

Сформируем далее вектор-столбец, совпадающий со вторым столбцом матрицы A

```
>>a = A(:,2)
   a=
      0
      2
      2
```

Разделитель «:» используется при формировании массивов. Выражение n:p:m означает список (диапазон) от n до m с шагом p (ис-

пользуемые здесь числа не обязательно целые, и список считается пустым, если приращение p не приближает к m, а отдаляет от него, например 1:-1:5). Выражение n:m означает частный случай при p=1. *Двоеточие без пределов означает весь промежуток изменения индекса* и позволяет сослаться на столбец A(:,i) или на строку A(j,:). Более того, команда A(:) превращает вектор-строку A в вектор-столбец, а матрицу A разворачивает по столбцам в один «длинный» вектор-столбец. Допустимы также и более причудливые массивы, представляющие соединение диапазонов и перечислений. Так, например, выражение [1,18.5:20.5,3,4:2:10,13] представляет собой сокращенную запись массива [1,18.5,19.5,20.5,3,4,6,8,10,13]. Такого рода объекты, но с целочисленными элементами, часто используются как множества индексов строк и столбцов при формировании массивов как части элементов другого массива.

Выполним теперь несколько умножений, используя для вывода результатов функцию disp. При таком способе вывода результатов идентификатор переменной не выводится, а в остальном результаты команд A и disp(A) совпадают.

```
>>A=a*a'; disp(A)
      0   0   0
      0   4   4
      0   4   4

>>b=a'*a; disp(b)
       8
```

Произведение матрицы размера [n r] на матрицу размера [r m] есть матрица размера [n m], именно поэтому здесь A – матрица, а b – скалярный квадрат вектора a. Следующее произведение является поэлементным

```
>> c=a.*a; disp(c')
       0     4     4
```

Переменные, создаваемые в течение сеанса, образуют рабочую область, представляющую совокупность данных, которыми пользователь оперирует, применяя средства системы Matlab. О текущем состоянии рабочей области пользователь имеет возможность фор-

мировать запросы. Так запрос >>whos заставляет систему дать подробный отчет

Name	Size	Elements	Bytes	Density	Complex
A	3 by 3	9	72	Full	No
B	3 by 2	6	48	Full	No
S	3 by 2	2	32	0.3333	No
a	3 by 1	3	24	Full	No
ans	2 by 3	6	48	Full	No
b	1 by 1	1	8	Full	No
c	3 by 1	3	24	Full	No

Grand total is 30 elements using 256 bytes

Как видно из вывода содержимого рабочей области, Matlab различает имена, набранные в разных регистрах, а векторы и скаляры трактует как матрицы размера [1 m], [n 1] и [1 1]. В последнем столбце сообщается, являются ли переменные комплексными, а в предпоследнем указывается доля заполненности. Матрицей разреженного типа в данном случае является заполненная на треть матрица S, которая, между прочим, по своему содержанию совпадает с полной матрицей B.

На запрос >>who сообщаются лишь имена переменных

Your variables are:
A S ans c
B a b

Все переменные или их часть могут быть сохранены на диске до следующих сеансов. А именно, по команде

save <имя файла без расширения> <переменные>

указанные переменные выгружается из рабочей области на диск в файл с указанным именем и с расширением mat. Переменные в списке разделяется пробелом и пишутся без скобок. *По умолчанию*, когда список переменных не указан, *сохраняются все переменные*, находящиеся в данный момент в рабочей области.

Обратно из файла в рабочую область данные загружаются командой load <имя файла без расширения>. Заметим, что *при загрузке с диска список переменных не указывается, так как система при записи переменных «запоминает» также и их идентификаторы*. Таким образом, после исполнения команды загрузки в рабочей области появляются переменные под теми же именами, под которыми их выгружали на диск. *По умолчанию,* когда имя файла не указано, *как при записи так и при чтении подразумевается служебный файл* matlab.mat, размещенный в корневом каталоге Matlab.

Ненужные переменные можно удалять из рабочей области. Команда clear <переменные> удаляет указанные переменные, а ее сокращенный вариант clear удаляет все переменные. В отличие от нее команда меню Edit – Clear Session лишь очищает командное окно, но отнюдь не удаляет переменные из рабочей области и не выбрасывает команды из стека. Заметим также, что исполнение любой команды при необходимости прерывается комбинацией клавиш <Ctrl+C>. Эта возможность используется, например, при зацикливании программы.

Matlab является открытой системой в двух смыслах. Во-первых, пользователь может пополнять ее внешними библиотеками m–файлов, в том числе собственными. Во-вторых, пользователь имеет возможность без ограничений читать все m–файлы системы (являющиеся исходными текстами программ), например, с целью знакомства с деталями языка или с авторской реализацией алгоритмов.

Если, например, при знакомстве с какой-либо программой с помощью системы DEMO краткие комментарии оказались недостаточными для выяснения, как именно данный фрагмент написан, пользователь всегда располагает возможностью обратиться к исходным файлам. Кнопка Info на панели системы DEMO вызовет информационный текст, в котором, в частности, содержится имя файла, реализующего демонстрируемую программу.

В заключение укажем две команды, которые помогут найти нужный файл. Команда what <имя каталога> выдает список файлов указанного каталога (по умолчанию – текущего), а команда which <имя функции/файла> выводит путь доступа к указанному объекту.

MATLAB В ПРИМЕРАХ

1. Решение системы линейных уравнений

Сформируем случайную матрицу A и случайную правую часть b, найдем решение системы Ax = b и оценим число обусловленности и норму невязки.

Решение в системе Matlab имеет вид

```
% Формирование случайных входных данных:
A=rand(10,10);  b=rand(10,1);

x=A\b;      % здесь левое деление
co=cond(A);
no=norm(A*x-b);
disp(['Число обусловленности = ',...
num2str(co)...
' Норма невязки = ', num2str(no) ]);
```

Здесь текст, следующий за символом «%», воспринимается как комментарий и на работе команды не отражается, а три точки в конце строки используются как знак продолжения оператора в следующей строке. Matlab имеет две встроенные операции матричного деления – левое $A\backslash b = A^{-1}b$ и правое $A/b = b\,A^{-1}$, а также несколько функций для вычисления норм векторов и матриц. *При большом значении числа обусловленности система предупреждает о возможной ненадежности результата.* Вот результат работы приведенного выше фрагмента

```
Число обусловленности = 47.87
Норма невязки = 5.223e-016.
```

Надо сказать, в системе Matlab имеется функция обращения матрицы inv(A), и решение могло быть таким: D = inv(A); x = D*b. Однако *всегда предпочтительнее пользоваться решателями* (/, \), так как в них анализируется специфика матрицы и по ней автоматически выбирается наиболее подходящий алгоритм. Кроме того, решатели используют (также автоматически выбираемые) масштабные коэффициенты для уравнений и неизвестных, позволяющие при ре-

шении линейной алгебраической системы минимизировать ошибки округления.

2. Решение нелинейного уравнения

Найти корень уравнения $4e^{-x^2} + 0.25(x - 3) = 0$.

Решение. Подготовим файл с именем fun0001.m, содержащий следующие три строки

```
function f = fun0001(x)
f =4 * exp(- x.^2) + (x - 3)/4;
end
```

Выполнив теперь команду поиска нуля функции x0 = fzero('fun0001',1), получим приближенное значение корня x0 = 0.7621. При обращении к функции, которой в качестве фактических параметров передаются имена других функций, как в данном примере, последние заключаются в апострофы. В качестве второго аргумента функции fzero выступает начальное приближение, в окрестности которого отыскивается решение уравнения fun0001(x)=0. Проверка y0 = fun0001(x0) дает результат y0 = 0.

В общем случае заголовок m–функции оформляется в виде: function [<выходн. парам.>]=<имя>(<входные парам.>), при этом *имя функции должно непременно совпадать с именем m–файла (без расширения), в котором она сохраняется. При единственном выходном параметре квадратные скобки опускаются.*

3. Поиск минимума

Найти минимум той же функции на отрезке [0, 3].

Решение. Выполнив команду xm = fmin('fun0001',0,3), получим результат xm = 1.5959. Функция fmin ищет на указанном отрезке локальный, а не глобальный минимум. Вычислим значение функции в точке минимума ym = fun0001(xm). Результат ym=- 0.2727. Изобразим график функции (и две точки, найденные в данной и предыдущей задачах) с помощью следующих команд

```
x=0:.1:3; y=fun0001(x);
% Установка белого фона в графическом окне:
whitebg;
   % Здесь одновременно три графика:
```

```
plot(x,y,'k',x0,y0,'ob',xm,ym,'xr');
title('Zero and minimum of the...
function fun0001');
xlabel('var  x');  ylabel('var  y')
grid on;    % сетка рисуется пунктиром
  % Изображение текста в позициях,
  % слегка отодвинутых от изображенных точек:
text(x0+.03,y0+.03,'(x0,y0);f(x0)=0');
text(xm–.22,ym+.04,'(xm,f(xm))')
```

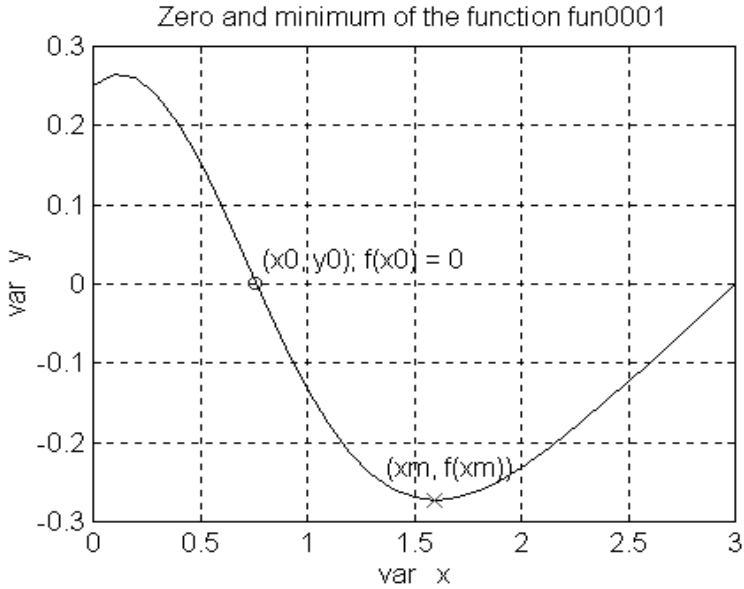


Рис.2. График функции и характерные точки

Здесь каждый третий аргумент функции plot задает спецификации изображения соответствующей функции: y(x) рисуется черным цветом (k – выделенная буква в слове black), пара (x0, y0) изображается голубым кружком, а пара (xm, ym) – красным косым крестиком. Более подробная информация содержится в специальном разделе, посвященном графике.

4. Аппроксимация табличных данных

Найти кубический полином, приближающий наилучшим образом в смысле метода наименьших квадратов функцию

$y(x) = 4e^{-x^2} + 0.25(x-3) + 0.03\sin(10x)$, $x = 0.2 \times k$, $k = 0,...,15$

Решение. Сформируем искомый полином $c_1x^3 + c_2x^2 + c_3x + c_4$ следующими тремя командами

```
x=0:.2:3; % значения от 0 до 3 с шагом 0.2
  % Здесь обращение к ранее определенной
  % функции fun0001:
y=fun0001(x)+.03*sin(10*x);
  % Коэффициенты полинома 3 степени:
coeff=polyfit(x,y,3)
```

Результатом является вектор-строка коэффициентов полинома

```
coeff = 0.0155  0.1113  -0.5761  0.3406.
```

5. Программная реализация задачи 4

Решение. Сформируем файл, содержащий программу аппроксимации данных и графического изображения результатов, и запишем его под именем prim002.m.

```
% begin of file prim002.m
x=0:.2:3;
y=exp(-x.^2)+(x-3)./4+0.03*sin(10*x);
c=polyfit(x,y,3);
txt=['coef:'];% начало сборки текста подписи
for l=1:4
  txt= [txt,' c',int2str(l),...
  ' = ',num2str(c(l))];
  % Функции int2str и num2str переводят
  % целое и вещественное числа
  % из двоичного представления в текстовое.
end   % конец цикла формирования подписи
% Значения на более детальной сетке:
x1=0:0.05:3; y1=polyval(c,x1);
% Табличные данные - зелеными кружочками:
plot(x,y,'og',x1,y1);
title('Data(o) and polynomial approximation');
xlabel(txt);
```

В данном случае мы имеем дело не с m–функцией, а с файлом сценария (script-файлом) prim002.m. Формальное отличие заключается в отсутствии заголовка и свободе выбора имени m–файла. Более глубокое различие состоит в том, что *у m–функции по умолчанию внутренние переменные локальны, а у файла сценария они глобальны*.

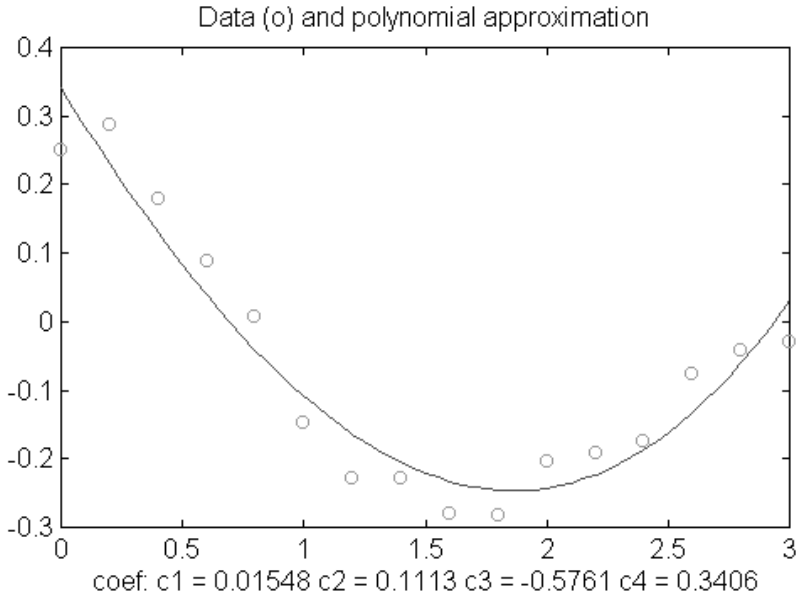


Рис.3. Аппроксимация табличных данных

Иначе говоря, после исполнения программы, оформленной как файл сценария, все ее переменные оказываются в рабочей области, а после работы m–функции сохраняются только выходные параметры и глобальные переменные. Заметим, что внутренние переменные m–функции при необходимости можно сделать глобальными, объявив их в операторе global.

Результат вызова программы prim002 изображен на рис.3.

6. Нахождение корней многочлена

Найти корни многочлена из предыдущего примера.

Решение – команда z = roots(c), где c – коэффициенты полинома. Результат – вектор-столбец (для краткости записанный как транспонированная строка), содержащий все корни

z = [-10.8194 2.9362 0.6928]'.

Вычисление значения полинома, например, в точке z(1): pv=polyval(c,z(1)) дает почти нуль: pv=1.4988e-015.

7. Задача интерполяции

Создать интерполяционный кубический полином по таблице значений (x = 0,1,2,3; y=1,2,1,3). Найти его значение в точке x = 2.5.
Решение .

```
x=0:3; y=[1 2 1 3];    % формирование таблицы
n=length(x)-1;
%  length возвращает длину вектора
coeff= polyfit(x, y, n);
y0= polyval(coeff, 2.5) % не подавляет вывода
```

Результат y0 = 1.3125.
Другие функции: interp1, interp2, interp3, griddata, spline.

8. Вычисление сумм и произведений

Вычислить сумму $S = \frac{1}{20}\sum_{j=1}^{20} \sin(j)$.

Решение
J = 1:20; S = .05*sum(sin(J))
Результат S = 0.9982.

Функция cumsum вычислила бы также все промежуточные значения. Для вычисления произведений служат функции prod и cumprod.

9. Вычисление интегралов

Вычислить $I = \int_{\pi}^{3\pi} \sin(x + \sqrt{x})\, dx$.

Решение. Создадим m–файл fquad.m, содержащий строки

```
function y=fquad(x)
```

```
y=sin(x+sqrt(x));
end
```

и выполним команду I=quad('fquad',pi,3*pi). Результат I = -0.6827.

Другие функции: quad8, trapz.

10. Решение задачи Коши для ОДУ

Решить уравнение $y''+ a\,y'+ b\,y = \ln(\cos(x)), 0 \le x \le 1$ при начальных данных $y(0) = 0.2,\ y'(0) = 0.5$ и значениях параметров $a = 1,\ b = 2$.

Решение. Заменой $u_1 = y,\ u_2 = y'$ приводим ОДУ к системе первого порядка $\mathbf{u}' = \mathbf{g}(x, \mathbf{u}),\ \mathbf{g} = (u_2,\ \ln(\cos(x)) - a\,u_2 - b\,u_1)'$ с начальными данными для вектора - столбца $\mathbf{u} = (0.2,\ 0.5)'$.

Сформируем m–функцию, вычисляющую правую часть системы ОДУ

```
function g=equat2(x,u)
%   EQUAT2(x,y) возвращает вектор-функцию правой
%   части ОДУ.  x - независимая переменная,
%   u=[u(1); u(2)]= [y; y'] - вектор-столбец
%   зависимой переменной

%   Виктор Паасонен, ИВТ СО РАН, март 2000
%
global a_gl b_gl;
g=[u(2); log(cos(x))-a_gl*u(2)-b_gl*u(1)];
end
```

и файл сценария, решающий систему ОДУ

```
% Решает систему u'=equat2(x,u)
% при частных значениях
% параметров a_gl=1, b_gl=2 функции EQUAT2,
% объявляемых здесь глобальными.
% Начальные данные u(1)=0.2, u1(1)=0.5.
%
a_gl=1.0; b_gl=2.0;
u0=.2; du0=.5;
% ОДУ  от 0 до 1 с начальными  данными
```

```
% [u0, du0] и точностью 1.e-04:
[x,u]=ode23('equat2',0.0,1.0,...
[u0,du0],1.e-04);
plot(x,u);
title('Solution by Runge & Kutta procedure');
```

Рисунок, полученный в результате работы данной программы, для краткости не приводится.

Другие функции: ode45.

Замечания

- В системе Matlab по команде help<имя программы> в командное окно выдаются все строки комментария до первого выполняемого оператора или до первой пустой строки, если таковая встретится раньше. *Именно на этом свойстве основан способ создания справки* для личных библиотек и отдельных программ. Так по команде help equat2 выдаются первые четыре строки комментария к функции equat2:

```
EQUAT2(x,y) возвращает вектор-функцию правой
части ОДУ.  x - независимая переменная,
u=[u(1); u(2)]= [y; y'] - вектор-столбец
зависимой переменной.
```

- Обычно функция «не видит» переменных сценария. Они становятся доступными, если в теле функции объявить их глобальными, как в данном примере. *Для глобальных переменных рекомендуется подбирать уникальные имена, например, с использованием символа подчерка*, иначе возможно их совпадение с именами, используемыми в системе.

- Функция ode23 (метод Рунге-Кутты) возвращает более одного параметра. *В таких случаях* и в заголовке функции и при обращении к ней *возвращаемые параметры* пишутся *в квадратных скобках* и разделяются запятыми.

- В последнем примере аргументы команды plot(x,u) имеют различные размерности: x –вектор, а u – матрица. В таких случаях

plot(x,u) изображает графики зависимости вектор-столбцов матрицы u как отдельных скалярных таблично заданных функций от x. Цвет каждого графика определяется циклически из шести фиксированных значений.

ОСОБЕННОСТИ СИСТЕМЫ MATLAB

Универсальность функций системы

Мы уже имели возможность убедиться в том, что арифметические операции в Matlab выполняется корректно для различных сочетаний типов операндов. Обычно допустимыми являются следующие сочетания:

- оба операнда –скаляры;
- один операнд – скаляр, другой – массив;
- оба операнда – массивы с согласованными размерностями.

Авторы системы следуют концепции, обеспечивающей универсальность (многозначность) не только простейших операций, но и всего множества функций Matlab. Так все элементарные математические функции допускают аргументы-массивы (результатом является, как правило, массив той же размерности). Для некоторых элементарных функций реализованы сугубо матричные их варианты. Они снабжены отличительным суффиксом m, например, expm и sqrtm вместо exp и sqrt. Для определяемых пользователем функций также возможна их матричная реализация.

Функции системы сделаны весьма естественно – разветвления алгоритмов в зависимости от числа входных и возвращаемых параметров и от их типов хорошо спроектированы и очень удобны для пользователя. Ниже приводится несколько наиболее ярких примеров такого рода.

- Поиск максимума. Для пары массивов равного размера max(X,Y) возвращает массив того же размера с поэлементным вычислением максимума на соответственных местах. Но функция max(X), если X вектор, возвращает наибольшую его компоненту, а если X матрица – то строку, содержащую максимумы по отдельным столбцам. Далее, обращение с двумя выходными параметрами [Y,I] = max(X) позволяет получить кроме строки максимумов по каждому из столбцов матрицы X еще и строку индексов, при которых максимумы достигаются. Последнее обращение работает и в случае, когда X вектор-строка или вектор-столбец. Наконец, если хотя бы один элемент аргумента в вы-

шеперечисленных обращениях оказался комплексным, максимум для всех элементов понимается как максимум модуля.

- Работа с диагоналями матрицы. Функция d=diag(X) извлекает главную диагональ матрицы X, а ее обобщенный вариант d=diag(X,k) извлекает побочную диагональ с заданным номером. Верхняя диагональ имеет положительный номер k, а нижняя – отрицательный, при этом значение abs(k) показывает, как далеко они отстоят от главной диагонали. Далее, X=diag(d) формирует квадратную матрицу X с вектором d на главной диагонали (все остальные элементы полагаются равными нулю), а X=diag(d,k) формирует квадратную матрицу порядка length(d) +abs(k) с вектором d на k-й диагонали и нулями на других местах.

- Сортировка данных. Функция Y=sort(X) в случае вектора упорядочивает его элементы по возрастанию. В случае матрицы упорядочиваются элементы каждого столбца. Функция [Y,I] = sort(X) возвращает дополнительно массив индексов, позволяющих позже восстановить исходный массив. Если аргумент содержит комплексные элементы, упорядочение ведется по модулям abs(X).

- Формирование массива случайных чисел. X=rand(n,m) формирует матрицу размера [n m], элементами которой являются случайные величины, равномерно распределенные на интервале (0,1). Выражение rand(n) – сокращенный вариант выражения rand(n,n). Функция rand без аргументов возвращает всего одно случайное число, а функция rand(size(A)) возвращает массив случайных чисел такого же размера, какой имеет массив A. Функция rand('seed') возвращает текущее значение базы генератора случайных чисел, а rand('seed',x0) устанавливает его равным x0. Для замены равномерного распределения нормальным распределением с нулевым математическим ожиданием и единичным среднеквадратичным отклонением в приведенных выше выражениях достаточно заменить функцию rand на randn.

- Поиск индексов массива по условию. Функция k=find(x) возвращает индексы ненулевых элементов вектора x. Функция [i,j]=find(X) возвращает пары индексов ненулевых элементов матрицы X. Обращение с тремя выходными параметрами [i,j,s]=find(X) позволяет получить в дополнение к индексам еще и сами значения ненулевых элементов. Если в приведенных выше обращениях заменить аргумент X некоторым логическим условием на него (например, X>4), функция find будет возвращать индексы элементов, удовлетворяющих сформулированному условию. Принято, что при обращении с условием [i,j,s]=find(X>4) параметр s будет булевым вектором, состоящим из одних единиц, а не вектором элементов матрицы X, удовлетворяющих указанному условию.

Приведенные примеры дают представление об обсуждаемом свойстве универсальности (гибкости и многозначности) функций Matlab. Не продолжая перечень примеров такого рода, заметим, что *почти все функции системы Matlab имеют универсальное действие*. Поэтому обращение к Справке относительно, казалось бы, знакомой функции позволяет иногда открыть для себя ее дополнительные возможности. Полезно помнить об этой особенности системы Matlab и не пренебрегать возможностью лишний раз щелкнуть по команде меню Help или набрать на клавиатуре команду help <имя функции>.

Организация пользовательских функций

Для создания собственных программ, универсальных в указанном выше смысле, особенно полезны функции nargin и nargout, возвращающие число входных и выходных аргументов, а также size, length, определяющие размеры, и функции issparse, ischar, isreal и другие, определяющие тип входных параметров.

Ниже приведен пример простейшей анимации, организованной путем последовательной активизации графических окон Для удобства в эту же программу «вмонтирована» возможность закрытия всех окон по окончании демонстрации. *Обратите внимание на оформление заголовка для функции, не возвращающей никаких параметров.*

Сами окна, содержащие изменяющуюся картинку, должны быть сформированы отдельно. Калейдоскопом картинок может быть, например, график функции, зависящей от параметра (различным значениям параметра соответствуют различные графические окна).

```
function showfig(n1,n2,k,pau)
%  SHOWFIG   Демонстрирует или закрывает
%                  последовательность графических окон.
%    ПРИМЕНЕНИЕ:
%  SHOWFIG(n1,n2,k,pau) - проводит (при k>0)
%    k циклов демонстрации окон с номерами
%    n1,...,n2 с паузой pau.
%  SHOWFIG(n1,n2,k) - проводит (при k>0) k
%    циклов демонстрации окон n1,...,n2 без
%    паузы.
%    При k=0 вместо демонстрации окон
%    осуществляется их закрытие.
%  SHOWFIG(n1,n2) - проводит один  цикл
%    демонстрации окон n1,...,n2.
%  SHOWFIG(n) - проводит один  цикл
%    демонстрации окон 1,...,n.

% установка входных значений по умолчанию
if nargin<4, pau=0; end
if nargin<3, k=1; end
if nargin<2, n2=n1; n1=1; end
if k<0|pau<0|n1>n2
   error('SHOWFIG: bad argument')
end

% Закрытие окон n1,…,n2:
    if k==0, delete(n1:n2); return; end
%  (оператор return заставляет немедленно
%  вернуться в вызывающую программу)

for ndemo = 1:k
   for nfig=n1:n2
      % Активизация окна с  указанным номером:
      figure(nfig);
     pause(pau);   % держит паузу pau (в сек)
 end          % конец  серии ndemo
end           % конец демонстрации
```

Предположим, в нашем распоряжении имеется 30 графических окон, каждое из которых содержит одно из состояний некоторого динамического процесса. Тогда при обработке команды showfig(30) с единственным входным аргументом функция nargin в теле функции showfig будет возвращать единицу. Поэтому выполнятся все установки по умолчанию, и результатом команды будет однократная демонстрация окон 1-30 с нулевой паузой. Быстро закрыть все эти окна удается по команде showfig(1,30,0).

Функция nargout используется аналогично. Ниже приводится пример функции, вычисляющей координаты центра, периметр и площадь прямоугольника (со сторонами, параллельными координатным осям) с заданными противоположными вершинами.

```
function  [rc,Per,s]=rectang(r0,r1)
%RECTANG  вычисляет координаты центра, периметр
%                и площадь прямоугольника.
%    ПРИМЕНЕНИЕ:
%  rc=rectang(r0,r1)возвращает только
%                координаты центра;
% [rc,Per]=rectang(r0,r1)– возвращает то же
%                плюс периметр Per;
% [rc,Per,s]=rectang(r0,r1)–  возвращает
%                еще и площадь s;
%
% Входные данные: r0,r1 – радиусы-векторы
% противоположных вершин (векторы длины 2).

If length(r0)~=2 | length(r1)~=2
error('RECTANG: bad length of argument'); end
rc=(r0+r1)/2;
if nargout>=2
   dr=r1-r0; Per=2*norm(dr,1); end
% Функция norm(x,p) в случае вектора x
% возвращает его p-норму, равную
% sum(abs(x).^p)^1/p для всех p~=0.
% Функция norm(A,p) в случае матрицы A
% возвращает подчиненную p-норму только
% в частных случаях p=1,2,inf.
if nargout==3, s=abs(dr(1)*dr(2)); end
```

Оформление функции с вызовом nargout позволяет избежать лишних вычислений. Например, в данном случае по команде c = restang([0 0],[1 2]) не будет вычисляться ни периметр, ни площадь, так как при таком вызове nargout = 1. Конечно, в данном примере это не очень важно, но при солидном объеме вычислений в отдельных ветвях программы такой подход может дать существенный выигрыш.

Специальные системные переменные

В системе Matlab имеются специальные переменные, скрытые в том смысле, что они не обнаруживаются запросами whos и who. Тем не менее, они всегда находятся в распоряжении пользователя, и их значения можно вывести, выполнив команду, состоящую только из идентификатора такой переменной. В таблице 2 приведены основные системные переменные.

Переполнение разрядной сетки при вычислениях сопровождается предупреждением, и соответствующая переменная получает значение inf, *но счет при этом не прерывается*. При операциях типа 0*inf, 0/0, inf/inf, inf-inf и других, приводящих к неопределенности, переменной присваивается так называемое нечисловое значение nan (no number). Переменная eps определяется как наименьшее положительное число, для которого еще выполняется неравенство 1+eps > 1. Например, для арифметики стандарта IEEE eps = 2^(-52)≈ 2.22e-016. Вещественные числа, превышающие realmax (≈ 1.8 e+308), заменяются числом inf, а числа, меньшие realmin (≈ 2.2 e-308), обнуляются.

Таблица 2

СИСТЕМНЫЕ ПЕРЕМЕННЫЕ MATLAB

ans	Результат последней операции, выполненной без явного присваивания
i	Мнимая единица
j	Мнимая единица
pi	Число π
inf	Бесконечность
nan	Неопределенность (нечисловое значение)
eps	Относительная машинная точность

realmax	Наибольшее вещественное число
realmin	Наименьшее вещественное число

Заметим, что для мнимой единицы зарезервированы две переменные. Причина состоит в том, что при использовании пары переменных для мнимой единицы выражения некоторых важных функций, используемых в физике и механике, выглядят более компактно, привычно и естественно, чем если бы использовалась только одна переменная.

Следует соблюдать осторожность при выборе имен переменных, так как *система формально не препятствует переопределению системных переменных* путем обыкновенного присваивания им каких угодно выражений. Особенно часто происходит переопределение пользователем переменных i, j, для использования их в качестве индексов, и eps в качестве малого параметра. В первом случае никаких последствий не будет, если только в сеансе не предполагается работа с комплексными числами, а во втором случае все вычисления, не относящиеся к целочисленной арифметике, станут проводиться с другой относительной точностью.

Функции минимизации по одной переменной fmin и по многим переменным fmins, а также функции пакета Optimization Toolbox, используют системный массив длины 18, имеющий идентификатор options. В нем для удобства собраны различные входные и выходные параметры, влияющие на работу указанных функций или характеризующие текущее состояние. В таблице 3 приводится описание некоторых наиболее часто используемых элементов этого массива.

Таблица 3

ПАРАМЕТРЫ ФУНКЦИЙ ОПТИМИЗАЦИИ

k	Смысл параметра options(k)
1	Признак вывода промежуточных результатов: true или false (значение по умолчнию)
2	Итерационная погрешность для аргумента, по умолчанию 1.e-4
3	Итерационная погрешность для функции, по умолчанию 1.e-4
10	Число выполненных итераций

14	Ограничение сверху для числа итераций, по умолчанию 500 для функции одной переменной и 100*k для функции k переменных

Все значения массива options можно вывести с помощью команды options = foptions.

Некоторые полезные функции

В настоящем кратком пособии невозможно описать все функции системы Matlab. Поэтому здесь дан беглый обзор только некоторых полезных функций, которые не упоминались в предшествующих разделах и не относятся к графической библиотеке – ей посвящен специальный раздел. Заметим, что хотя здесь даны только основные формы вызова, для большинства приведенных ниже функций характерна универсальность, т.е. существует несколько форм обращения к ним.

□ *Решение системы линейных уравнений.*

- ✓ tril(A), triu(A) – формирование нижней и верхней треугольных матриц;
- ✓ chol(A), lu(A), qr(A) – разложение Холецкого, LU и QR разложения матриц;
- ✓ x=nnls(A,b) – решение системы Ax = b методом наименьших квадратов;
- ✓ eig(A) – определение собственных векторов и собственных значений матрицы A;
- ✓ svd(A) – сингулярное разложение матрицы A.

□ *Трассировка и отладка программ.*

- ✓ keyboard – остановка выполнения программы и передача управления в режим ввода команд с клавиатуры. В этом режиме локальные переменные приостановленной программы полностью доступны, т.е. их содержимое можно вывести или изменить путем присвоения им любых допустимых выражений. Выполнение программы возобновляется по команде return;
- ✓ cputime – время работы процессора в секундах, прошедшее с момента запуска Matlab. Используется, например, для оценки

времени, требуемого для выполнения программы или ее отдельных фрагментов;

- ✓ flops – выдает число выполненных операций с плавающей запятой. Используется наряду с cputime для исследования объема вычислений во фрагментах программы.

□ *Функции ввода – вывода.*

- ✓ x = input('<приглашение>') – ввод значения/значений с клавиатуры по приглашению с присвоением его/их переменной x. Тип переменной и ее размер определяется системой автоматически в зависимости от того, что именно вводится;
- ✓ [x,y] = ginput – ввод координат мышью. По данной команде открывается графическое окно с квадратной рабочей областью и в ней появляется перекрестие курсора мыши. По каждому щелчку левой кнопкой массивы x,y пополняются текущими координатами курсора. Ввод завершается по нажатию правой кнопки мыши;
- ✓ fprintf – форматирование сообщений, выводимых в командное окно;
- ✓ sprintf – запись форматированных данных в виде строки;
- ✓ diary <имя файла> – (дневник) перехватывает выводы, обычно осуществляемые в командное окно, и направляет их в указанный файл. Эта переадресация выводов может проводиться выборочно путем включения записи (diary on) и ее выключения (diary off), создавая файл-отчет в нужной форме.

□ *Оценка типа и характера переменных.*

- ✓ isglobal, isinf, isnan и т.д. (такого рода функций довольно много) – проверяют, является ли переменная глобальной, бесконечной, неопределенной и т.д.;
- ✓ eval('<выражение>') – заставляет систему воспринимать текст выражения как команду или часть выражения;
- ✓ feval ('<имя функции>',x1,…, xn) – применяется для вычисления внешней функции с входными параметрами x1, …, xn, если ее имя передается данной программе как формальный параметр. Если провести аналогию с Фортраном, можно сказать, что в системе Matlab функция feval применяется в вызываемых программах для вычисления тех функций, которые на языке Фор-

тран требуется описывать как внешние в операторе EXTERNAL;

- ✓ y=all(x) – возвращает 1, если все элементы вектора x не равны нулю, и 0 в противном случае. Для аргументов-матриц функция определяется более сложно;
- ✓ y=any(x) – возвращает 1, если хотя бы один элемент вектора x не равен нулю, и 0 в противном случае. Для аргументов-матриц функция определяется более сложно;

□ *Формирование матриц.*

- ✓ zeros(m,n) – формирование нулевой матрицы;
- ✓ ones(m,n) – формирование матрицы из единиц;
- ✓ eye(m,n) – формирование единичной матрицы;
- ✓ fliplr(A), flipud(A) и rot90(A,k) – повороты матрицы A относительно вертикали, горизонтали и k-кратный ее поворот на прямой угол;
- ✓ reshape(A,m,n) – переформирование матрицы A, заключающееся в разворачивании ее по столбцам в один вектор с последующим «нарезанием» столбцов новой матрицы в соответствии с указанным размером;
- ✓ vander – формирование матрицы Вандермонда (имеется много других функций для создания специальных матриц).

□ *Разреженные матрицы.*

- ✓ A=full(S) – преобразование разреженной матрицы в полную;
- ✓ B=spdiags(S,d), S=spdiags(B,d) – извлечение диагоналей разреженной матрицы и, наоборот, формирование разреженной матрицы по заданным диагоналям;
- ✓ find(X), find(<условие на X>) – определение массива индексов положительных элементов переменой X или элементов переменой X, удовлетворяющих указанному условию;
- ✓ spfun('fun',S) – вычисление функции fun только для ненулевых элементов разреженной матрицы S (сильно сокращает объем вычислений при большой размерности S и малой ее заполненности). Функция fun – это имя m–файла, содержащего вычисляемую функцию, входным аргументом которой является матрица S. Переменная fun может быть также именем встроенной функции. Заметим, что A=spfun('exp',S) и B=exp(S) не одно и то же,

так как в первом случае для нулевых элементов вычисления не производятся, и на соответствующих местах матрицы A также сгенерируются нули, а во втором случае на этих же местах появятся единицы (exp(0)=1);

- ✓ spones(S)– формирование матрицы связности, т.е. матрицы с единицами на тех местах, где элементы матрицы S не являются нулевыми;
- ✓ speye(m,n) – формирование единичной разреженной матрицы;
- ✓ normest(S), condest(S) – оценка нормы и числа обусловленности разреженной матрицы;

□ *Преобразование систем координат.*

- ✓ cart2sph, cart2pol – преобразование декартовых координат в сферические, цилиндрические и полярные;
- ✓ sph2cart, pol2cart – обратные преобразования.

□ *Анализ Фурье, статистика, специальные функции.*

- ✓ fft,fft2,ifft,ifft2 – Одномерное и двумерное прямые и обратные дискретные преобразования Фурье;
- ✓ mean, std, cov – вычисление среднего значения, стандартного отклонения, матрицы ковариаций;
- ✓ besselj, beta, ellipj, gamma – некоторые специальные функции.

ГРАФИЧЕСКИЕ ФУНКЦИИ

Элементы дескрипторной графики

Графические элементы в системе Matlab (окна, графики, поверхности, оси и другие), а также управляющие элементы графического интерфейса пользователя (командные кнопки, линейки прокрутки, окна редактирования, метки и другие) представляют собой объекты, наделенные атрибутами или свойствами (тип, размеры, цвет и т.д.). Каждый из атрибутов может принимать значения из допустимого множества, и их совокупность определяет вид конкретного объекта. Графические элементы идентифицируются дескриптором как уникальным числовым именем. Например, дескриптором графического окна является его номер, фигурирующий в заголовке окна.

Функции, осуществляющие построение графических объектов, возвращают дескриптор, что всегда дает возможность получить имена-идентификаторы объектов. Например, если вместо plot(...) выполнить команду p = plot(...), то в наше распоряжение поступает кроме рисунка еще и идентификатор p, значение которого и есть его дескриптор. Выполнение команды plot(...) также связывает с рисунком значение дескриптора, однако оно скрыто от пользователя, и в этом случае рисунок не получает идентификатора.

Тем не менее, значение дескриптора активного графического объекта всегда можно запросить. Его возвращает функция gco (сокращение от get current object). Так команда p=gco обеспечивает активный объект идентификатором p. Этот второй путь задания идентификатора эквивалентен первому, если только активен тот же самый рисунок. Для активизации графического объекта (осей, графика, текста подписи и т.д.) достаточно щелкнуть мышью на нем. Заметим, что щелчок достаточно близко к границе графического окна сделает активным само окно, а не рисунок в окне, и тогда функция gco возвратит номер графического окна.

Если в окне имеется несколько рисунков (например, построенных с помощью команды subplot), щелчок по выбранному рисунку активизирует именно его, хотя сама по себе активизация объекта внешне никак не проявляется. После активизации рисунка можно менять его свойства. В частности, команды colormap, axis и другие

меняют значения атрибутов активного объекта, что приводит к его немедленной перерисовке с учетом новых свойств. Изменение размеров окна также вызывает его перерисовку. Отметим также полезную функцию H=findobj('Атрибут','Значение'), которая находит объекты, указанный атрибут которых имеет заданное значение.

Некоторые функции возвращают не единственный дескриптор. Так команда [C, H]= contour(…), подготавливающая массив данных C для изображения линий уровня функций, возвращает также вектор дескрипторов H, компоненты которого соответствуют отдельным линиям уровня. Через компоненты вектора H пользователь получает доступ к редактированию атрибутов каждой линии уровня в отдельности.

Свойства объекта можно изменять, используя функцию set. Команда set (H, 'Атр1', Знач1, 'Атр2', Знач2,…) устанавливает для графического объекта с дескриптором H (кратко говоря, для объекта H) значения указанных атрибутов. Например, команда

```
set(H, 'Position', [10 10 200 250])
```

изменяет положение и размеры окна H, а команда

```
set(gcf, 'DefaultTextColor', 'RED')
```

устанавливает для текстовых объектов в текущем графическом окне красный цвет в качестве значения по умолчанию. Здесь использована функция gcf, аналогичная gco, возвращающая дескриптор активного графического окна (сокращение от get current figure). Команда set(H, 'Атрибут') отображает для объекта H все допустимые значения данного атрибута, а команда set(H) отображает все имена атрибутов объекта H и для каждого из атрибутов все их допустимые значения.

Команда V = get(H, 'Атрибут') возвращает значение атрибута для объекта H, а get (H) отображает все имена атрибутов для него и их текущие значения. Команда H = gca возвращает дескриптор текущей системы координат, являющейся отдельным от поля рисунка объектом.

Команда reset(H) сбрасывает все атрибуты H к их значениям по умолчанию, за исключением позиции. Например, reset(gca) сбрасывает нестандартные атрибуты текущей системы координат.

Элементы управления

К графическим элементам управления относятся командные кнопки, линейки прокрутки, окна редактирования, метки и другие

объекты. Тип графического элемента управления определяется списком его атрибутов. Функция

H = uicontrol ('Атр1', Знач1, 'Атр2', Знач2,…)

создает в текущем окне графический элемент управления с заданными свойствами и возвращает его дескриптор. Атрибуты элементов управления можно так же, как и в случае графиков, запросить командой get и установить командой set.

Пользовательские меню создаются аналогично. Функция

H = uimenu ('Атр1', Знач1, 'Атр2', Знач2,…)

создает элемент меню в строке меню текущего графического окна и возвращает его дескриптор. Команда uimenu(H,...) создает новое меню с объектом H в качестве предка. Если H – дескриптор окна, создается элемент меню, а если H – дескриптор меню, создается пункт меню. Если же H – дескриптор пункта меню, создается выпадающее меню. Такова иерархия пользовательских меню в системе Matlab.

Атрибуты uicontrol, uimenu и других объектов могут быть установлены как во время создания объекта с использованием пар параметров <ИмяАтрибута / Значение>, так и изменены впоследствии командой set.

Для связывания элементов меню и графических элементов управления с определенными действиями, обычно оформленными в виде m–функции, необходимо для атрибута 'CallBack' установить в качестве значения строку вызова этой m–функции.

Двумерные графики

Построение графика в линейном масштабе по осям осуществляет функция plot. При обращении plot(y) с одним параметром предполагается, что независимая переменная принимает натуральные значения 1,...,length(y). При двух параметрах функция plot(x,y) строит один (если x и y – векторы) или несколько (если y или оба аргумента – матрицы) графиков с независимой переменной x. В случае указания трех переменных plot(x,y,s) строковая переменная s определяет способ отображения функции y(x) и, по соглашению, принятому в Matlab, может содержать не более трех символов таблицы 4.

Таблица 4

СПОСОБЫ ОТОБРАЖЕНИЯ ГРАФИКА

Тип линии		Тип точки		Цвет	
Сплошная	-	Точка	.	Желтый	y
Штриховая	- -	Плюс	+	Фиолетовый	m
Двойной пунктир	:	Звездочка	*	Голубой	c
Штрих-пунктир	-.	Кружок	o	Красный	r
		Крестик	x	Зеленый	g
				Синий	b
				Белый	w
				Черный	k

По умолчанию график изображается сплошной линией. Если способ отображения не указан, при построении всех графиков для данного обращения цвета выбираются циклически из первых шести значений, указанных в таблице 4.

Команда plot(x1,y1,s1,x2,y2,s2,…) позволяет объединить несколько графиков функций в одной системе координат. В частности, этой формой пользуются в тех случаях, когда при построении графика y(x) трех символов переменной s оказывается недостаточно для описания способа отображения. Так, например, команда plot(x,y,'--c',x,y,'og') рисует график функции y(x) голубой пунктирной линией, а точки (x,y), по которым он строится, отображает зелеными кружочками.

Функции semilogx, semilogy или loglog аналогичны функции plot. Они используют логарифмический масштаб по x, по y или по обеим переменным соответственно. Функция fplot('<имя функции>',a,b) строит график с автоматически вычисляемым наиболее целесообразным неравномерным шагом, отслеживая локальные особенности в поведении функции. Функция bar(x,y,<тип линии>) строит столбцовую диаграмму, а stairs(x,y) – ступенчатый график.

Имеется еще ряд графических функций:

- polar – график в полярных координатах;
- hist – гистограмма;
- rose – гистограмма в полярных координатах;
- stem – дискретный график;
- errorbar – график с указанием интервала ошибок;

- comet – «живое» изображение траектории движения точки;
- compass – график в виде стрелок, исходящих из начала координат.

Изображение функций в пространстве

Функция plot3 предназначена для построения точек и линий в пространстве. По своему синтаксису она аналогична функции plot, необходимо лишь заменить двумерную таблицу (x,y) тройкой массивов (x,y,z), представляющих пространственные координаты последовательности точек. Функция comet3 (аналог comet) рисует траекторию движения точки, рассчитываемую по заданной таблице координат (x,y,z) точек в пространстве.

Функция [X,Y] = meshgrid(x,y) формирует двумерные массивы X,Y координат регулярной двумерной сетки по заданным одномерным сеткам x и y. Эта команда очень часто используется для последующего поэлементного вычисления значения функции на регулярной сетке. Отметим ее сокращенный вариант meshgrid(x), соответствующий одинаковым аргументам y = x.

Функция quiver(X,Y,DX,DY) рисует векторное поле, определяемое величиной и направлением значений вектора (DX,DY), заданных для массива точек плоскости (X,Y). Предусмотрена также возможность масштабирования длин стрелок векторного поля и выбора атрибутов линии.

Функция contour(x,y,Z,n) рисует n автоматически выбираемых линий уровня функции Z(x,y). По умолчанию принято n=10, а замена четвертого аргумента n вектором v позволяет отказаться от автоматического выбора уровней, рисуя их для значений, заданных элементами вектора v. Ниже приводится программа построения линий уровня и векторного поля градиента функции, определенной в квадратной области изменения (x,y).

```
whitebg;    % переключение на белый фон
h=pi/20;    % шаг сетки
[X,Y]=meshgrid(-10*h:h:10*h);
Z = sin(X).*exp(-Y.^2-X.^2);
[DX,DY] = gradient(Z,h,h);
contour(X,Y,Z,'k');
hold on;
quiver(X,Y,DX,DY,1.5,'k');
```

```
whitebg;    % возврат черного фона
```

Следует иметь в виду, что функция whitebg работает как переключатель, возвращая при повторном вызове черный фон. Поэтому во избежание неприятностей нужно либо применять парный вызов whitebg, как в данном фрагменте, либо вообще не включать его в программу, а загружать однократно из командного окна или из файла startup. В противном случае результатом четных попыток исполнения программы будет черный квадрат.

Система располагает возможностью маркировки линий уровня значениями функции на них. Для этого требуется лишь применить несколько иную форму обращения, которая возвращает так называемую матрицу описания линий уровня C=contour(…), состоящую из двух строк

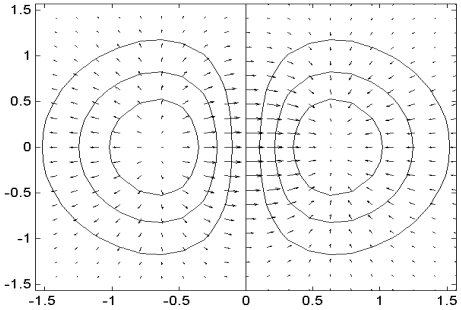


Рис.4. Линии уровня и векторное поле градиента функции.

$$C = [c1\ \ x1_1\ x1_2\ \ldots\ x1_{k1}\ \ c2\ \ x2_1\ \ x2_2\ \ldots\ x2_{k2}\ \ldots\ ;$$
$$k1\ \ y1_1\ y1_2\ \ldots\ \ y1_{k1}\ \ k2\ \ y2_1\ \ y2_2\ \ \ldots\ y2_{k2}\ \ldots\ ].$$

Здесь вещественное число c1 в начале первой строки матрицы C представляет собой значение функции на первой линии уровня, натуральное число k1 в начале второй строки означает число точек на ней, а следующие за ними массивы чисел $x1_1, x1_2, \ldots, x1_{k1}$ и $y1_1, y1_2, \ldots, y1_{k1}$ представляют собой координаты этих точек. Затем размещается аналогичный блок данных, определяющий вторую линию уровня (значение функции c2 на втором уровне, число точек на нем k2 и их координаты), и так для каждой линии уровня.

Когда матрица описания уровней C сформирована, с помощью одной из команд clabel(C,v) или clabel(C,'manual') осуществляется маркировка уровней. Первая маркирует линии уровня автоматически, а наиболее интересная вторая команда предоставляет возможность маркировки мышью. Нажатие левой кнопки мыши в выбранном месте линии уровня устанавливает очередной маркер, а нажа-

тие правой кнопки мыши заканчивает маркировку. Заметим, что с помощью команды contour3 линии уровня могут строиться также и в пространстве. Следующий фрагмент строит несколько изображений функции двух переменных.

```
[X,Y]=meshgrid([-2:.2:2]);
Z=X.*exp(-X.^2-Y.^2); colormap('gray');
subplot(221),contour3(X,Y,Z,11,'r');
title('contour3');
subplot(222),mesh(X,Y,Z);title('mesh');
subplot(223),meshz(X,Y,Z);title('meshz');
axis('off');
subplot(224),meshc(X,Y,Z);title('meshc');
axis('off');
```

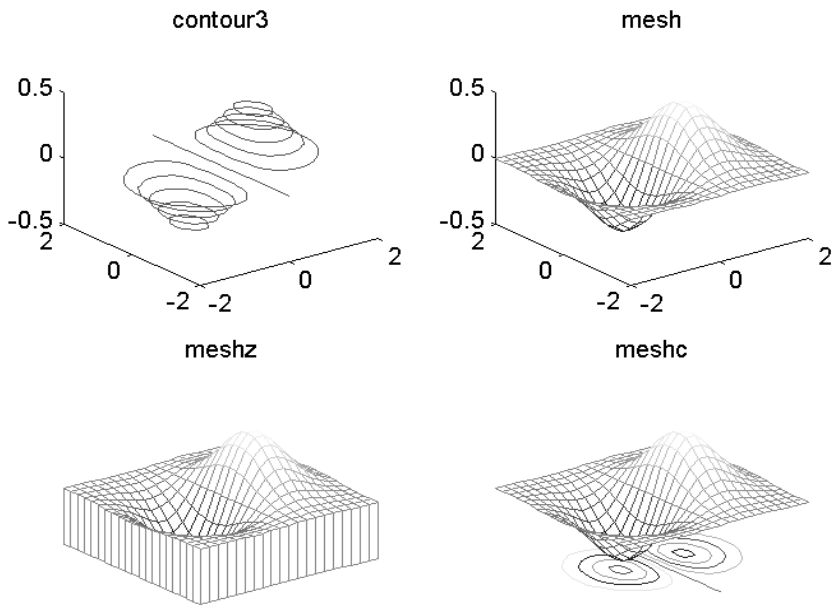


Рис.5. Изображения функции двух переменных.

Сетчатые поверхности рисует функция mesh и ее модификации. На рис.5. представлено несколько изображений одной функции. Разбиение графического окна на несколько подокон для размещения нескольких изображений реализуется с помощью команды

subplot('nmq') или subplot(n,m,q). Числа n и m указывают, на сколько равных частей делится полное графическое окно по вертикали и по горизонтали, а q означает номер выбираемой части из числа n□m полученных таким образом подокон. Обратите внимание, что команда axis('off') делает невидимыми оси координат.

Следующая программа демонстрирует различные способы построения поверхностей, первый из которых waterfall дает вариант сетчатой поверхности, у которой отсутствует одно из семейств кривых, образующих поверхность, что и создает аналогию с водопадом, а последующие три способа на базе функции surf реализуют различные схемы затенения поверхности командой shading.

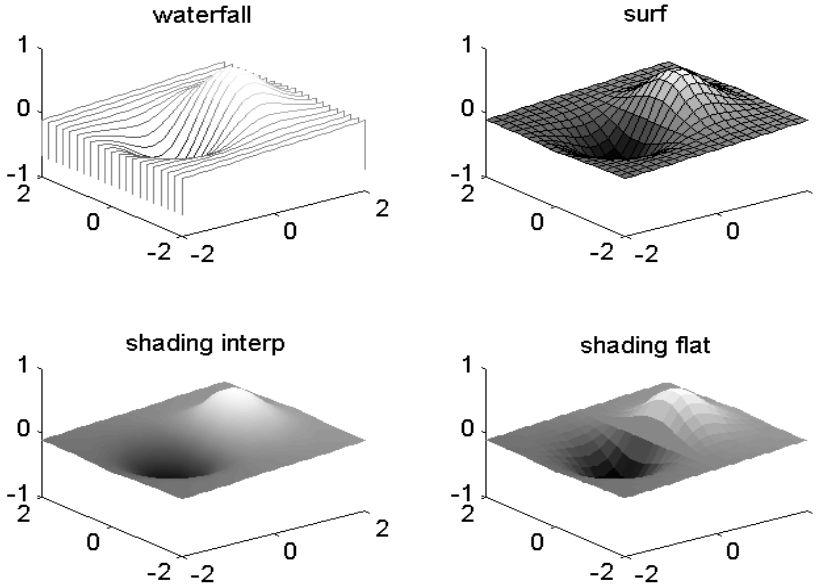


Рис.6. Различные виды поверхностей

```
[X,Y]=meshgrid([-2:.2:2]);
Z=2*sin(pi/4*X).*cos(pi/4*Y) .*...
exp(-X.^2-Y.^2)+.05*X; colormap('gray');
subplot(221),waterfall(X,Y,Z);
title('waterfall');
subplot(222),surf(X,Y,Z);title('surf');
subplot(223),surf(X,Y,Z);shading interp;
```

```
title('shading interp');
subplot(224),surf(X,Y,Z);shading flat;
title('shading flat');
```

Допустимым является также и вызов surf(x,y,Z), где x и y – одномерные массивы сеток по отдельным переменным. При вызове surf с дополнительным четвертым параметром C реализуется более общее цветовое решение, при котором цвета ячеек определяются массивом C. Цвет ребер по умолчанию черный (в формате смеси трех цветов [r g b] = [0 0 0]), а в общем случае определяется значением атрибута EdgeColor. При задании EdgeColor = none ребра вообще не рисуются.

Способ рисования граней определяется атрибутом FaceColor. Для сетчатых поверхностей его значение всегда совпадает с цветом фона, а для затененных поверхностей принимает значения flat или interp, устанавливаемые функцией shading. В первом случае цвет одинаков для всей грани, а во втором случае он интерполируется по значениям цвета в углах грани. Команда shading faceted восстанавливает состояние по умолчанию. Заметим, что команда surfc так же, как и meshc, дополнила бы рисунок проекциями линий уровня на плоскость (x,y).

Matlab позволяет с хорошим качеством имитировать подсветку, используя функцию surfl. В общем виде функция зависит от пяти аргументов surfl(X,Y,Z,s,k), где вектор s задает направление на источник света, а k определяет модель подсветки. Вектор s задается своими компонентами либо в декартовых координатах s=[sx, sy, sz] либо в сферических s=[az,elev]. По умолчанию принят второй способ с азимутом az=-37.5□ и возвышением elev=30□. Четырехкомпонентный вектор k включает в себя параметры, учитывающие эффекты отраженного света, диффузного отражения, зеркального отражения и зеркального распространения. По умолчанию k = [0.55 0.6 0.4 10].

На рисунке 10 изображена подсвеченная затененная поверхность, построенная с помощью фрагмента

```
surfl(X,Y,Z);
colormap(gray);
shading interp;
axis('off');
```

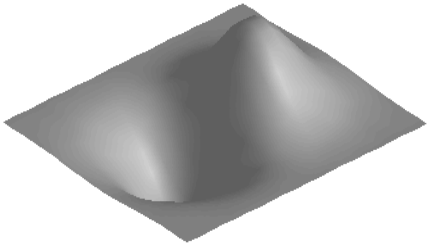


Рис.10. Поверхность с подсветкой

Когда значения функции заданы не на регулярной сетке, а на произвольном множестве точек, непосредственно воспользоваться функциями mesh и surf невозможно. В этом случае оказывается полезной функция griddata, интерполирующая функцию двух переменных с одного произвольного множества точек на другое. Ниже приводится программа, демонстрирующая эту возможность. Результат работы программы изображен на рис. 11.

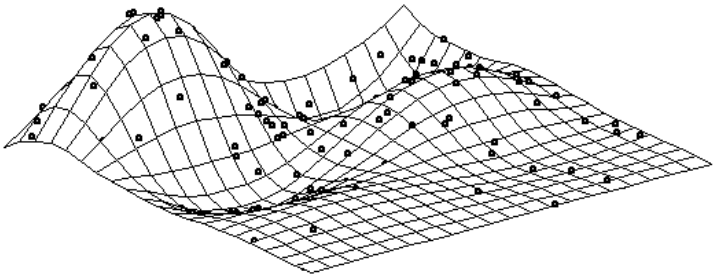


Рис.11. Интерполяция данных с нерегулярной сетки

```
% Нерегулярное (случайное)множество точек:
   x = rand(1,200); y = rand(1,200);
% Функция z(x,y) на ней:
   z = sin(3*pi*y).*y.^2.*sin(2*pi*x);
% Регулярная сетка:
  t = 0:.05:1; [X,Y] = meshgrid(t,t);
% Интерполированные значения функции на ней:
```

```
  Z = griddata(x,y,z,X,Y);
% Картинка
  mesh(X,Y,Z);
  colormap([0 0 0]);
  hold on; plot3(x,y,z,'ko');
  axis('off');
  disp(' Press any key to continue');pause;
% Перерисовка крошечными кружочками
  p = findobj('LineStyle','o');
  set(p,'MarkerSize',[3],'LineWidth',[1.5]);
```

В последних двух строках данной программы отыскивается объект, обладающий заданным признаком, и меняется его облик. Между прочим, *допускается сокращение наименований атрибутов (а также и их значений, если они выражены строкой) до трех символов*, если только это не приводит к неопределенности. Например, последние две строки можно было записать в виде

```
p = findobj('LineS','o');
set(p,'Mar',[3],'LineW',[1.5]),
```

однако в данном случае атрибуты 'LineStyle' и 'LineWidth' менее чем пятью символами записать невозможно, так как первые четыре символа в именах совпадают; это как раз и привело бы к неопределенности. *Такого рода неопределенность Matlab контролирует, выдает сообщение об ошибке и прерывает работу.*

Область определения изображаемой функции не обязана быть прямоугольной. Следующая программа представляет пример изображения функции, определенной в круге.

```
R = (0:.2:2)';          % Значения радиуса
phi = 2*pi*[0:50]/50;   % Значения угла
% Координаты полярной сетки в осях x,y:
X = r * cos(phi);Y = r * sin(phi);
% Значения функции на ней:
Z = exp(-(X+Y).^2).*sin(X*pi);
% Ее изображение:
meshz(X,Y,Z); colormap([0 0 0]);
```

Результат представлен на рисунке 12. Если в приведенной программе ограничить диапазон изменения угловой переменной, заменив вторую строку командой phi=2*pi*[0:35]/50, то получится поверхность, изображающую ту же функцию, но определенную не в круге, а в секторе (рис.13).

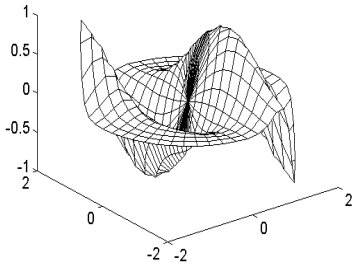


Рис.12. Функция, определенная в круге

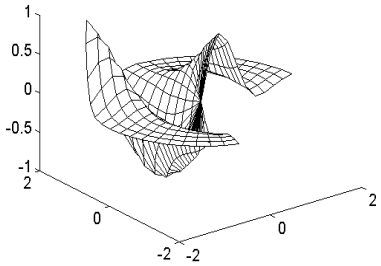


Рис.13. Функция, определенная в секторе

Аналогично строятся сетчатые и затененные поверхности для функций, определенных на «криволинейных четырехугольниках».

Замечания общего характера

Matlab достаточно хорошо автоматически выбирает масштабы по осям координат. Однако в некоторых случаях (например, при визуализации функций, зависящих от параметра) от автоматического выбора целесообразно отказаться, так как автоматически выбираемый масштаб, скорее всего, будет изменяться с изменением параметра, а поведение функции в переменном масштабе наблюдать сложно.

Управление масштабами осей осуществляет функция axis. Например, команда axis([x0 x1 y0 y1 z0 z1]) устанавливает принудительно указанные масштабы (для осей на плоскости аргументом будет вектор из четырех компонент), а команда axis('auto') восстанавливает автоматический режим. Команда w = axis возвращает вектор масштабов активного графика, а команды axis('ij') axis('xy') устанавливают начало отсчета в левый верхний и левый нижний угол окна соответственно (это так называемые матричные и декартовы координаты). Имеется еще несколько форм вызова функции axis, уста-

навливающих, например, равные расстояния между метками на осях axis('equal') или одинаковые диапазоны по осям axis('square') и другие.

Непосредственное отношение к масштабу имеет также и функция zoom. Команда zoom on включает режим масштабирования активного графического окна. Нажатие левой (правой) кнопки мыши увеличивает (уменьшает) масштаб рисунка вдвое в окрестности курсора мыши. Для указания иного масштаба следует выделить нужную область окна, удерживая левую кнопку мыши. Команда zoom off выключает режим масштабирования, команда zoom без параметров работает как переключатель режимов. Наконец, команда zoom out возвращает график в первоначальное состояние.

В дополнение к рассмотренным ранее функциям отображения надписей на рисунках (title, xlabel, ylabel, zlabel, clabel, text), отметим еще функцию legend, которая формирует легенду графика, а также очень полезную функцию gtext('text'), которая с помощью мыши позволяет поместить заданный текст в любую, заранее не определенную позицию.

Положение точки наблюдения также регулируется. Команда view([az ev]) устанавливает ее по заданному азимуту и углу возвышения, а команда view([x y z]) – по декартовым координатам. Заметим, что первый способ соответствует наблюдению из бесконечности и формирует изображение без перспективы. При втором способе формируется изображение в перспективе, характер которой зависит от выбора точки наблюдения.

Палитра цветов определяется функцией colormap(C). В общем случае C – матрица размера m□3, каждая строка которой C(q,:) = [r(q) g(q) b(q)]указывает доли красного, зеленого и синего цветов. Величина m характеризует степень детальности палитры. В частности, если m=1, то изображение является однотонным. Текущее значение матрицы C возвращает команда C = colormap.

Некоторые палитры являются стандартными в системе Matlab, имеют специальные имена и могут активизироваться командой colormap(<имя>). Это цветовые схемы gray, white, cool, hot, bone, copper, flag, hsv, jet, pink и prism. Их проще увидеть, чем описать. Для этого достаточно изобразить любой рисунок и поочередно активизировать различные палитры. Команда colormap('default') воз-

вращает пользователя к принятой по умолчанию схеме hsv (hue–saturation–value), соответствующей цветам радуги.

Matlab поддерживает технологию копирования через буфер обмена содержимого графических окон в документы других приложений Windows. Если при этом в дальнейшем планируется создание черно-белых твердых копий, содержащих рисунки Matlab, то следует помнить, что при стандартном черном фоне для этой цели подходят только палитры gray и white. При переключении на белый фон (whitebg) вместо палитры white, которая даст на белом фоне невидимое изображение, необходимо установить черную палитру командой colormap([0 0 0]).

Matlab допускает также непосредственную печать рисунков по команде print из программы или командного окна, а также по команде меню графического окна. Полезно помнить, что при печати изображения на черно-белом принтере по команде меню print происходит обращение цвета, т.е. рисунок в палитре white на бумаге будет выглядеть естественно – черным на белом фоне.

ДОПОЛНИТЕЛЬНЫЕ ЗАДАЧИ

Сглаживание функций

Предположим, что функция f=y+δ, заданная таблицей значений на неравномерной сетке, есть сумма истинных значений функции и случайной ошибки.

Применим известный способ сглаживания, устраняющий высокочастотные составляющие ошибки, схема которого приведена на рис.8. Суть метода состоит в параметрическом осреднении текущего значения B и значения D, полученного интерполяцией соседних значений A и C.

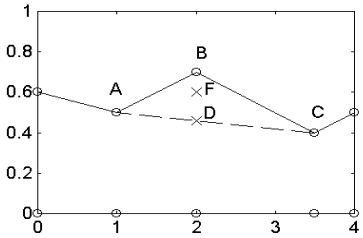


Рис.14. Схема сглаживания

В зависимости от выбора значения параметра осреднения новое значение F располагается ближе к B (слабое сглаживание) или ближе к D (радикальное неустойчивое сглаживание). Ниже приводится текст соответствующей m–функции.

```
    function w=level1(u,x,a)
    % LEVEL1 сглаживает функцию u(x),
    % содержащую случайные ошибки
    % ПРИМЕНЕНИЕ:  w=level1(u,x,a)
    %              w=level1(u,x)
    % x - вектор значений независимой переменной
    % u - вектор (или матрица) значений функции
    % Для матриц сглаживание по строкам, размер
    % строк матрицы u должен совпадать с длиной x
    % a - (0<a<1) параметр  (по умолчанию 1/4)
    % w - сглаженная функция
    %
    %   Виктор Паасонен, сентябрь 1997
    %
[n m] = size(u);
if nargin<3, a = 1/4; end
    % Вектор конечных разностей по x:
dx = diff(x);
```

```
  % Матрица конечных разностей w по строкам
w = u; dw = (diff(w'))';
  % Сглаживание
for i = 2:m–1
    w(:,i)=w(:,i)+a*(dw(:,i)*dx(i-1)...
    -dw(:,i-1)*dx(i))/(dx(i)+dx(i-1));
end
```

«Странный» способ вычисления конечных разностей функции w объясняется спецификой функции diff, действующей по столбцам, а не по строкам. Для скалярной функции w примененное ухищрение излишне. Следующий фрагмент демонстрирует применение программы level1 для сглаживания данных, содержащих «шум».

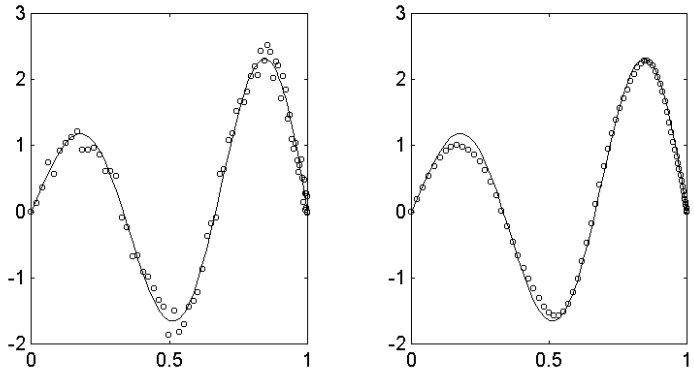


Рис.15. Сглаживание случайных ошибок функции f(x)

```
p=0:pi/150:pi/2; epsil=.5;
    % x - неравномерная сетка:
x=sin(p);
  % Исходная гладкая функция:
f0=exp(x).*sin(3*pi*x);
    n=length(x); I=2:n-1;
    % Функция с ошибкой (с «шумом»)
f=f0; f(I)=f(I)+(rand(size(I))-0.5)*epsil;
subplot(1,2,1),  h0=plot(x,f0,'k');
set(h0, 'LineWidth', [1]);  hold on;
h=plot(x,f,'ok');  set(h, 'MarkerSize', [4]);
    %  Функция set устанавливает
    %  нестандартные значения атрибутов
```

```
for k=1:40     % многократное сглаживание
    f = level1(f,x,1/8);
    subplot(1,2,2),   h = plot(x,f,'ok');
    set(h, 'MarkerSize', [4]);
    axis([0 1 -2 3]); pause(0);
    % pause(0) обеспечивает показ
    % промежуточных состояний,
    % без этого оператора они будут утеряны
end
    hold on; h0 = plot(x,f0,'k');
    set(h0, 'LineWidth', [1]);
```

В первом фрагменте программы формируется неравномерная сетка, моделируется функция и после этого «истинные» значения (сплошная линия) и значения, содержащие ошибку (кружки), изображаются в левой части рис.15. Во втором фрагменте проводится многократное сглаживание с выводом в правую половину рис.15 изображений промежуточных состояний. В результате кружки постепенно перестраиваются во все менее хаотичную последовательность. К заключительной картинке снова добавляется «истинная» кривая.

Функция level1 написана так, чтобы ею можно было воспользоваться и в случае векторных функций. Это делает ее пригодной также и для функций двух переменных. Ниже приведена программа и результат сглаживания поверхности.

```
y=-pi/2:pi/40:pi/2;        %  сетка по y
x=sin(y(1:2:length(y)) );  %  сетка по x
 % Координаты сеточной области:
[X,Y]=meshgrid(x,y);
    % Исходная функция:
f0=exp(-X.^2-Y.^2);
  % Массивы индексов внутренних точек:
n=length(x); I=2:n-1;
m=length(y); J=2:m–1;
  % Функция с ошибкой во внутренних узлах:
f=f0; f(J,I)=f(J,I)+(rand(m–2,n-2)-0.5)*.2;
clf; colormap(gray); % очистка окна
subplot(1,3,1), surfl(x,y,f0);
v=axis;          % запоминание параметров осей
```

```
subplot(1,3,2), surfl(x,y,f);
axis(v);              % установка таких же осей
for k=1:5
  f=level1(f,x);   % простое сглаживание по x
    g=f';  g=level1(g,y);
    f=g';            % сглаживание по y с транспонированием
     %  Сглаженная k раз функция:
    subplot(1,3,3), surfl(x,y,f);
    axis(v);  pause(0);
end
```

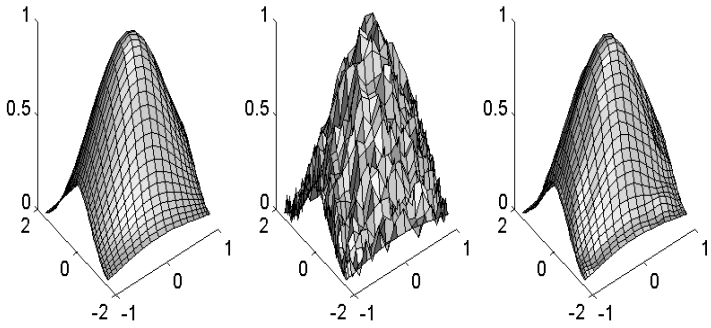


Рис.16. Исходная, возмущенная и сглаженная поверхности.

Один цикл сглаживания состоит в последовательном сглаживании каждой строки по x, а затем – каждого столбца по y. Очевидно, сглаживание по столбцу эквивалентно сглаживанию по строке транспонированного массива с последующим обратным транспонированием.

Неявная разностная схема с весами

Следующая программа реализует расчет краевой задачи для уравнения теплопроводности $U_t=U_{xx}$ на отрезке [0,5]. На обеих границах поддерживается нулевая температура U=0, а начальная температура представляет нагретое «пятно» внутри отрезка:

$U_0(x)=100\ (x-1)^4(x-2)^4$ на отрезке $1<x<2$;
$U_0(x)=0$ вне отрезка $1<x<2$.

Расчет ведется с помощью неявной схемы с весами

$$(E - b\tau\Lambda)\frac{u^{n+1} - u^n}{\tau} = \Lambda u^n \ ,$$

$$\Lambda u_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} \ .$$

```
k=101; h=5/(k-1);  tau=.5;   % шаги по x и t
b=.51;                  % вес схемы b
m=k-2;                  % число внутренних точек;
r=tau^2/h; c=-r*b;      %  вспомогательные параметры;
i=1:k;                  %  индексы всех точек
ip=2:k-1;               %  индексы внутренних точек;
t=0; x=i*h-h;           %  счетчик времени,сетка по x;
    % Начальная температура,
    % отличная от нуля только на отрезке (1,2):
u0=zeros(1,k);
J=find((1<x)&(x<2));  % индексы точек пятна
u0(J) = 100*(x(J)-1).^4 .*(x(J)-2).^4;
  % Рисунок U(x,0):
plot(x,u0,'k'), axis([0 5 0 .5]);
pause; hold on;

    % Формирование разреженной матрицы системы:
e = ones(m,1);
A=spdiags( [c*r*e  1-2*c*r*e  c*r*e],...
-1:1, m, m);

while t<5,  t=t+tau;      %  цикл по времени
    g= (r*diff(u0,2))';    %  правая часть схемы
    y=A\g;  %  вектор разности y=U(t)-U(t-tau)
    u0(ip)=u0(ip)+y';    %  решение U(t)
    plot(x,u0,'k'); pause(0);   % рисунок U(x,t)
end
```

Центральным пунктом этой программы является функция spdiags, формирующая разреженную матрицу из заданных диагоналей, описание которой ввиду ее особой важности приведено ниже. Заметим, что для ленточных матриц алгоритм решения системы с использованием решателей (/, \) нисколько не медленнее, чем прогон-

ка, если только матрица системы формируется как разреженная. Однако и в случае произвольных, не ленточных матриц, когда прогонка уже не применима, решатели для разреженных матриц нормально действуют и оказываются более эффективными, чем решатели для полных матриц.

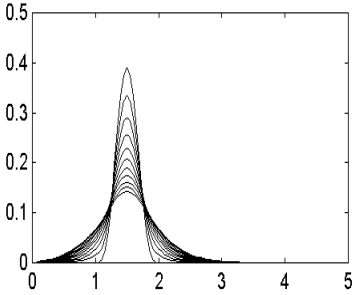


Рис.17. Растекание пятна.

Примененная здесь функция Matlab spdiags обобщает встроенную функцию diag и имеет дело с тремя массивами, которые могут быть как входными, так и выходными параметрами. Вот эти массивы:

- A – m □ n матрица, обычно разреженная, с отличными от нуля элементами, размещенными на p диагоналях. Знак номера диагонали определяет, находится она выше главной диагонали или ниже, а модуль показывает удаленность от нее;
- B – матрица размера min(m, n) □ p, обычно полная, столбцы которой являются диагоналями A;
- d – вектор длины p, целочисленные компоненты которого определяют номера диагоналей в матрице A.

Возможны четыре различных действия при обращении к функции spdiags:

- Извлечение всех диагоналей матрицы A, отличных от нуля: [B, d] = spdiags(A);
- Извлечение заданных диагоналей матрицы A (с номерами, заданными вектором d): B = spdiags(A,d);
- Замена заданных диагоналей новыми, представленными столбцами матрицы B: A = spdiags(B,d,A);
- Создание разреженной матрицы A размера m □n из диагоналей, заданных в B: A = spdiags(B,d,m,n);

Связь между A, B и d проще всего сформулировать в виде условного оператора:

```
if m >= n
    for k = 1:p
      for j = max(1,1+d(k)):min(n,m+d(k))
        B(j,k) = A(j-d(k),j);
      end
    end
else
  for k = 1:p
      for i = max(1,1-d(k)):min(m,n-d(k))
        B(i,k) = A(i,i+d(k));
      end
    end
end
```

Некоторые элементы B, соответствуя позициям «вне» A, не определены этими циклами. *Они не используются, когда B входной параметр, и принудительно обнуляются, когда B выводится.*

Например, прямоугольная разреженная матрица

```
A = [11    0    13    0    0
      0   22     0   24    0
      0    0    33    0   35
     41    0     0   44    0
      0   52     0    0   55
      0    0    63    0    0
      0    0     0   74    0 ]
```

имеет параметры m = 7, n = 5 и p = 3. Заметим, что ненулевые диагонали не являются в данном примере соседними, а имеют номера -3, 0 и 2.

Команда [B,d] = spdiags(A) возвращает матрицу

```
B = [ 41   11    0
      52   22    0
      63   33   13
      74   44   24
       0   55   35 ],
```

столбцы которой представляют собой диагонали A, и вектор номеров ненулевых диагоналей

```
d = [ -3
       0
       2 ].
```

Обратите внимание, что короткие диагонали с номерами -3 и 2, согласно принятому правилу формирования B, дополнены нулями.

И, наоборот, выражение spdiags(B,d,7,5), где B и d – полученные выше массивы, возвращает нас к оригиналу A. При этом замена нулей в матрице B любыми другими числами не отразилась бы на результате последней команды, так как согласно правилу элементы B, выступающие за габариты A, игнорируются.

Если разреженная матрица структурно отличается от «многодиагональной», ее можно формировать в два этапа: регулярную часть – с помощью функции spdiags, а остальные элементы – иначе, например, простым или циклическим присваиванием.

ЛИТЕРАТУРА

1. Moler C., Herskovitz S., Little J., Bangert S. *MATLAB for Macintosh Computers User`s Guide*. The MathWorks, Ins. 1987.
2. Потемкин В.Г. *Система MATLAB. Справочное пособие*.– М., ДИАЛОГМИФИ, 1997. –370с.
3. Потемкин В.Г., Рудаков П.И. *MATLAB 5 для студентов. Изд. второе.* М., ДИАЛОГМИФИ, 1999. –448с.
4. Потемкин В.Г. *Система инженерных и научных расчетов MATLAB 5.x.* Т1. М., ДИАЛОГМИФИ, 1999. –368с.
5. Потемкин В.Г. *Система инженерных и научных расчетов MATLAB 5.x.* Т2. М., ДИАЛОГМИФИ, 1999. –304с.
6. Медведев В.С., Потемкин В.Г. *Control System Toolbox. MATLAB 5 для студентов.* М., ДИАЛОГМИФИ, 1999. –288с.
7. Гультяев А.К. *MATLAB 5.2. Имитационное моделирование в среде Windows.* С.-П., Корона-принт, 1999. 288с.
8. Dongarra J.J., Bunch J.R., Moler C.V., Stewart G.V. *LINPACK User`s Guide.* Philadelphia, 1979.
9. Goleman T.F., C. Van Loan. *Handbook for Matrix Computation.* SIAM, Philadelphia, 1988.
10. *Signal Processing Toolbox User`s Guide.* Natick: MathWorks, Ins., 1993.
11. Gilben J., Moler C., Schreiber R. *Sparse Matrices in MATLAB: Design and Implementation* //SIAM Jornal on Matrix Analysis and Applications. 1992. V3. P.333-356.
12. Marcus. M. *Matrices and Matlab.* – Prentice Hall, 1993.

СОДЕРЖАНИЕ

Паасонен Виктор Иванович

ИНСТРУМЕНТ НАУЧНЫХ ИССЛЕДОВАНИЙ MATLAB

Учебное пособие

Подписано в печать 2000 г. Уч.-изд. л. 3.95.
Офсетная печать. Формат 60 x 84 1/16.
Тираж 100 экз. Заказ № .

Лицензия ЛР №021285 от 6 мая 1998 г.

Издательский центр НГУ;
630090, Новосибирск, ул. Пирогова, 2.