АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ
КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И КРИПТОГРАФИИ

Методическое пособие

ВВЕДЕНИЕ В СИСТЕМУ MATLAB

по курсу «Математические пакеты в решении инженерных задач»
для студентов специальности
200900 «Сети связи и системы коммутации»

Астрахань
2004

Разработано: Водолазская И.В., к.ф.-м.н., доцент кафедры «Прикладная математика и криптография»

Рецензент: зав. кафедрой АСОиУ, к.т.н., доц. Лаптев В.В.

Цель пособия: пособие представляет собой руководство к выполнению лабораторных работ по курсу «Математические пакеты в решении инженерных задач». Каждая лабораторная работа содержит краткое описание методов вычислений, примеры, порядок выполнения лабораторной работы, задания, контрольные вопросы.

Практическое пособие утверждено на заседании кафедры ПМиК
Протокол N от “ “ 2004 г.

Содержание

Предисловие

MATLAB – система многоцелевого назначения, которая вышла на рынок программных продуктов почти двадцать лет назад и с тех пор непрерывно совершенствовалась. Но первоначально ее основу составляли алгоритмы решения систем линейных уравнений и задач на собственные значения, откуда и произошло ее название «матричная лаборатория». Теперь она представляется наиболее эффективной при проведении прикидочных расчетов и при разработке новых алгоритмов. Сейчас уже существует несколько десятков специальных приложений к MATLAB'у, посвященных более узким проблемам. Это обработка сигналов и изображений, инженерное программирование в виде блок-схем, решение экономических задач и многое другое. Но любое из этих приложений можно изучать только после первоначального освоения MATLAB'а.

В методическое пособие включены две лабораторные работы. Предполагается, что студенты уже имеют базовые знания по математическому анализу (дифференциальное и интегральное исчисление, решение дифференциальных уравнений), комплексным числам, векторам и матрицам. При выполнении первой лабораторной работы студенты осваивают стандартные программные структуры и команды MATLAB'а: числа, матрицы, функции, графическое представление функций, действия с массивами. При выполнении второй лабораторной работы студенты получают навыки решения следующих задач: нахождение нулей и точек экстремума функции, вычисление определенных интегралов, решение дифференциальных уравнений, а также знакомятся с возможностями пакета символьных вычислений Symbolic Math Toolbox.

Лабораторная работа 1

Основы работы с MATLAB

Среда MATLAB включает интерпретатор команд на языке высокого уровня, графическую систему, пакеты расширений и реализована на языке С. Вся работа организуется через командное окно (*Command Window*), которое появляется при запуске программы matlab.exe. В процессе работы данные располагаются в памяти (*Workspace*), для изображения кривых, поверхностей и других графиков создаются графические окна.

В командном окне в режиме диалога проводятся вычисления. Пользователь вводит команды или запускает на выполнение файлы с текстами на языке MATLAB. Интерпретатор обрабатывает введенное и выдает результаты: числовые и строковые данные, предупреждения и сообщения об ошибках. *Строка ввода* помечена знаком >>. В командном окне показываются вводимые с клавиатуры числа, переменные, а также результаты вычислений. Имена переменных должны начинаться с буквы. Знак = соответствует операции присваивания. Нажатие клавиши *Enter* заставляет систему вычислить выражение и показать результат. Наберите с клавиатуры в строке ввода:

```
» a=2+51-37
```

Нажмите клавишу *Enter*, на экране в *зоне просмотра* появится результат вычисления:

```
a = 16
```

Все значения переменных, вычисленные в течение текущего сеанса работы, сохраняются в специально зарезервированной области памяти компьютера, называемой *рабочим пространством системы* MATLAB *(Workspace).* Командой *clc* можно стереть содержимое командного окна, однако это не затронет содержимого рабочего пространства. Когда исчезает необходимость в хранении ряда переменных в текущем сеансе работы, их можно стереть из памяти компьютера командой *clear* или *clear(имя1, имя2,* …). Первая команда удаляет из

памяти все переменные, а вторая – переменные с именами *имя1* и *имя2*. Командой *who* можно вывести список всех переменных, входящих в данный момент в рабочее пространство системы. Для просмотра значения любой переменной из текущего рабочего пространства системы достаточно набрать ее имя и нажать клавишу *Enter.*

После окончания сеанса работы с системой MATLAB все ранее вычисленные переменные теряются. Чтобы сохранить в файле на диске компьютера содержимое рабочего пространства системы MATLAB, нужно выполнить команду меню *File* | *Save Workspace As …* .По умолчанию расширение имени файла *mat,* поэтому такие файлы принято называть MAT-*файлами*. Для загрузки в память компьютера ранее сохраненного на диске рабочего пространства нужно выполнить команду меню:

File | *Load Workspace … .*

Вещественные числа и тип данных double

Система MATLAB представляет на машинном уровне все действительные числа заданные мантиссой и показателем степени, например, 2.85093E+11, где буквой E обозначается основание степени равное 10. Этот основной тип данных носит название *double.* MATLAB по умолчанию использует формат *short* для вывода вещественных чисел, при котором показываются только четыре десятичных цифры после запятой.

Введите с клавиатуры пример:

```
» res=5.345*2.868/3.14-99.455+1.274
```

Получите результат вычисления:

```
res =  -93.2990
```

Если требуется полное представление вещественного числа *res*, введите с клавиатуры команду:

```
» format long
```

и далее наберите имя переменной

```
» res
```

нажмите клавишу *Enter* и получите более подробную информацию*:*

```
res = -93.29900636942675
```

Теперь все результаты вычислений будут показываться с такой высокой точностью в течение данного сеанса работы в среде системы MATLAB. Если требуется до прекращения текущего сеанса работы вернуться к старой точности визуального представления вещественных чисел в командном окне, нужно ввести и исполнить (нажав клавишу *Enter*) команду:

```
» format short
```

Целые числа показываются системой в командном окне в виде целых чисел.

Над вещественными числами и переменными типа *double* производятся арифметические операции: сложения +, вычитания -, умножения *, деления / и возведения в степень ^ . Приоритет в выполнении арифметических операций обычный. Операции одинакового приоритета выполняются в порядке слева направо, но круглые скобки могут изменить этот порядок.

Если нет необходимости видеть в командном окне результат вычисления некоторого выражения, то в конце введенного выражения следует поставить точку с запятой и только после этого нажать *Enter.*

В системе MATLAB присутствуют все основные элементарные функции для вычислений с вещественными числами. Любая функция характеризуется своим именем, списком входных аргументов (перечисляются через запятую и стоят внутри круглых скобок, следующих за именем функции) и вычисляемым (возвращаемым) значением. Список всех имеющихся в системе элементарных математических функций может быть получен по команде *help elfun*. В Приложении 1 перечислены стандартные функции вещественного аргумента.

Вычислите выражение, включающее вычисление функции арксинус*:*

```
» 2*asin(1)
```

Убедитесь, что получился следующий результат:

```
ans = 3.1416,
```

соответствующее числу «пи». В системе MATLAB для вычисления числа «пи» есть специальное обозначение: *pi.* (Список системных переменных MATLAB находится в Приложении 2).

MATLAB имеет также логические функции, функции, связанные с целочисленной арифметикой (округления до ближайшего целого: *round*, усечение дробной части числа: *fix*). Есть еще функция *mod* – остаток от деления с учетом знака, *sign* – знак числа, *lcm* – наименьшее общее кратное, *perms* – вычисление числа перестановок и *nchoosek* – числа сочетаний и много других. Многие из функций имеют область определения, отличную от множества всех действительных чисел.

Помимо арифметических операций над операндами типа *double* выполняются еще операции отношения и логические операции. Операции отношения сравнивают между собой два операнда по величине. Эти операции записываются следующими знаками или комбинациями знаков (Таблица 1):

Таблица 1

Символьные обозначения операций отношения

<	<=	>	>=	~=	==
Меньше	Меньше или равно	Больше	Больше или равно	Не равно	Равно

В случае истинности операции отношения ее величина равна 1, а в случае ложности – 0. Операции отношения имеют более низкий приоритет, чем арифметические операции.

Наберите с клавиатуры выражение с операциями отношения и вычислите его:

```
» a=1; b=2; c=3;
» res=(a<b)+(c~=b)+(b==a)
```

Вы получите следующий результат:

```
res = 2
```

Логические операции над вещественными числами обозначаются знаками, перечисленными в таблице 2:

Таблица 2

Символьные обозначения логических операций

&	|	~
И	ИЛИ	НЕ

Первые две из этих операций являются бинарными (двухоперандными), а последняя – унарной (однооперандной). Логические операции трактуют свои операнды как «истинные» (не равные нулю) или «ложные» (равные нулю). Если оба операнда операции «И» истинны (не равны нулю), то результат этой операции равен 1 («истина»); во всех остальных случаях операция «И» вырабатывает значение 0 («ложь»). Операция «ИЛИ» вырабатывает 0 («ложь») только в случае, когда являются ложными (равными нулю) оба операнда. Операция «НЕ» инвертирует «ложь» на «истину». Логические операции имеют самый низкий приоритет.

Комплексные числа и комплексные функции

Комплексные переменные, как и вещественные автоматически имеют тип *double* и не требуют никакого предварительного описания. Для записи мнимой единицы зарезервированы буквы i или j. В случае, когда коэффициентом перед мнимой единицей является не число, а переменная, между ними следует обязательно использовать знак умножения. Итак, комплексные числа можно записывать следующим образом:

```
» 2+3i; -6.789+0.834e-2*i; 4-2j; x+y*i;
```

Почти все элементарные функции допускают вычисления с комплексными аргументами. Вычислите выражение:

```
» res=sin(2+3i)*atan(4i)/(1-6i)
```

Получится результат:

```
-1.8009 - 1.9190i
```

Специально для работы с комплексными числами предназначены следующие функции: *abs* (абсолютное значение комплексного числа), *conj* (комплексно сопряженное число), *imag* (мнимая часть комплексного числа), *real* (действительная часть комплексного числа), *angle* (аргумент комплексного числа), *isreal* («истина», если число действительное). Функции комплексного переменного перечислены в Приложении 1.

В отношении арифметических операций ничего нового для комплексных чисел (по сравнению с вещественными) сказать невозможно. То же самое относится и к операциям отношения «равно» и «не равно». Остальные операции отношения вырабатывают результат исходя только из действительных частей этих операндов.

Введите выражение, получите результат и объясните его:

```
» c=2+3i; d=2i;
» c>d
```

Логические операции трактуют операнды как ложные, если они равны нулю. Если же у комплексного операнда не равна нулю хотя бы одна его часть (вещественная или мнимая), то такой операнд трактуется как истинный.

Числовые массивы

Для создания одномерного массива можно использовать операцию конкатенации, которая обозначается с помощью квадратных скобок [ ]. Элементы массива помещаются между скобками и отделяются друг от друга пробелом или запятой:

```
» al=[1 2 3]; d=[1+2i,2+3i,3-7i];
```

Для доступа к индивидуальному элементу массива нужно применить операцию индексации, для чего после имени элемента указать в круглых скобках индекс элемента.

Можно изменять элементы уже сформированного массива путем применения операций индексации и присваивания. Например, введя:

```
» al(3)=789;
```

мы изменим третий элемент массива. Или, после введения:

```
» al(2)=(al(1)+al(3))/2;
```

второй элемент массива станет равным среднему арифметическому первого и третьего элементов. Запись несуществующего элемента вполне допустима – она означает добавление нового элемента к уже существующему массиву:

```
» al(4)=7;
```

Применяя после выполнения этой операции к массиву *a1* функцию *length*, находим, что количество элементов в массиве возросло до четырех*:*

```
» length(al)
ans = 4
```

Тоже самое действие – «удлинение массива *a1*» - можно выполнить и с помощью операции конкатенации*:*

```
» al=[al 7];
```

Можно задать массив, прописывая все его элементы по отдельности:

```
» a3(1)=67; a3(2)=7.8; a3(3)=0.017;
```

Однако этот способ создания не является эффективным.

Еще один способ создания одномерного массива основан на применении специальной функции, обозначаемой двоеточием (операция формирования диапазона числовых значений). Через двоеточие следует набрать первое число диапазона, шаг (приращение) и конечное число диапазона. Например:

```
» diap=3.7:0.3:8.974;
```

Если не нужно выводить на экран весь получившийся массив, то в конце набора (после конечного числа диапазона) следует набрать точку с запятой. Чтобы узнать, сколько элементов в массиве, следует вызвать функцию *length (имя массива)*.

Для создания двумерного массива (матрицы) также можно использовать операцию конкатенацию. Элементы массива набираются один за другим со-

гласно их расположению в строках, в качестве разделителя строк используется точка с запятой.

Введите с клавиатуры:

```
» a=[1 2; 3 4; 5 6]
```

Нажмите *ENTER,* получим:

```
a =
    1    2
    3    4
    5    6
```

Полученную матрицу *a* размером 3x2 (первым указывается число строк, вторым – число столбцов) можно сформировать также вертикальной конкатенацией вектор-строк:

```
» a=[[1 2];[3 4];[5 6]];
```

или горизонтальной конкатенацией вектор-столбцов:

```
» a=[[1;3;5],[2;4;6]];
```

Структуру созданных массивов можно узнать с помощью команды *whos(имя массива)*, размерность массива – функцией *ndims*, а размер массива – *size.*

Двумерные массивы можно задать также с помощью операции индексации, прописывая по отдельности его элементы. Номер строки и столбца, на пересечении которых находится задаваемый элемент массива, указываются через запятую в круглых скобках. Например:

```
» a(1,1)=1; a(1,2)=2; a(2,1)=3;
» a(2,2)=4; a(3,1)=5; a(3,2)=6;
```

Однако будет намного эффективнее, если до начала прописывания элементов массива, создать массив нужного размера функциями *ones (m,n)* или *zeros(m,n),* заполненный единицами или нулями (m – число строк, n – число столбцов). При вызове этих функций предварительно выделяется память под заданный размер массива, после этого постепенное прописывание элементов нужными

значениями не требует перестройки структуры памяти, отведенной под массив. Использование этих функций возможно и при задании массивов других размерностей.

Если после формирования массива *X* потребуется, не изменяя элементов массива, изменить его размеры, можно воспользоваться функцией *reshape (X, M, N),* где *M и N* – новые размеры массива *X*

Объяснить работу этой функции можно, только исходя из способа, каким система MATLAB хранит элементы массивов в памяти компьютера. Она хранит их в непрерывной области памяти упорядоченно по столбцам: сначала располагаются элементы первого столбца, вслед за ними расположены элементы второго столбца и т.д. Помимо собственно данных (элементов массива) в памяти компьютера хранится также управляющая информация: тип массива (например, *double*), размерность и размер массива, другая служебная информация. Этой информации достаточно для определения границ столбцов. Отсюда следует, что для переформирования матрицы функцией *reshape* достаточно изменить только служебную информацию и не трогать собственные данные.

Поменять местами строки матрицы с ее столбцам можно операцией транспортирования, которая обозначается знаком .' (точка и апостроф). Например,

```
» A=[1 1 1; 2 2 2; 3 3 3];
» B=A.'
B =
    1    2    3
    1    2    3
    1    2    3
```

Операция ' (апостроф) выполняет транспонирование для вещественных матриц и транспонирование с одновременным комплексным сопряжением для комплексных матриц.

Объекты, с которыми работает MATLAB, являются массивами. Даже одно заданное число во внутреннем представлении MATLAB является массивом,

состоящим из одного элемента. MATLAB позволяет делать вычисления с огромными массивами чисел также легко как и с одиночными числами, и это является одним из самых заметных и важных преимуществ системы MATLAB над другими программными пакетами, ориентированными на вычисления и программирование. Помимо памяти, необходимой для хранения числовых элементов (по 8 байт на каждый в случае вещественных чисел и по 16 байт в случае комплексных чисел), MATLAB автоматически при создании массивов выделяет еще и память для управляющей информации.

Вычисления с массивами

В традиционных языках программирования вычисления с массивами осуществляются поэлементно в том смысле, что нужно запрограммировать каждую отдельную операцию над отдельным элементом массива. В М-языке системы MATLAB допускаются мощные групповые операции над всем массивом сразу. Именно групповые операции системы MATLAB позволяют чрезвычайно компактно задавать выражения, при вычислении которых реально выполняется гигантский объем работы.

Операции сложения и вычитания матриц (знакомые вам из линейной алгебры) обозначаются стандартными знаками + и -.

Задайте матрицы *A* и *B* и выполните операцию сложения матриц:

```
» A=[1 1 1; 2 2 2; 3 3 3]; B=[0 0 0; 7 7 7; 1 2 3];
» A+B
```

Если используются операнды разных размеров, выдается сообщение об ошибке, за исключением случая, когда один из операндов является скаляром. При выполнении операции *A* + *скаляр* (*A* – матрица) система расширит *скаляр* до массива размера *A*, который и складывается далее поэлементно с *A*.

```
» A+5
ans = 6    6    6
      7    7    7
      8    8    8
```

Для поэлементного перемножения и поэлементного деления массивов одинаковых размеров, а также поэлементного возведения в степень массивов, применяются операции, обозначаемые комбинациями двух символов: **.*** , **./,** и **.^**. Использование комбинаций символов объясняется тем, что символами * и / обозначены специальные операции линейной алгебры над векторами и матрицами.

Кроме операции ./, называемой операцией правого поэлементного деления, есть еще операция левого поэлементного деления .\. Объясним разницу между этими операциями. Выражение A ./ B приводит к матрице с элементами *A (k, m) /B (k, m)*, а выражение A .\ B приводит к матрице с элементами *B (k, m) /A (k, m)*.

Знак * закреплен за перемножением матриц и векторов в *смысле линейной алгебры*.

Знак \ закреплен в системе MATLAB за решением довольно сложной задачи линейной алгебры – нахождением корней системы линейных уравнений. Например, если требуется решить систему линейных уравнений

$$Ay = b,$$

где *A* – заданная квадратная матрица размера *N* x *N*, *b* – заданный вектор-столбец длины *N*, то для нахождения неизвестного вектор-столбца *y* достаточно вычислить выражение *A* \ *b* (это равносильно операции: $A^{-1} \cdot B$).

Типичные задачи аналитической геометрии в пространстве, связанные с нахождением длин векторов и углов между ними, с вычислением скалярного и векторного произведений, легко решаются разнообразными средствами системы MATLAB. Например, для нахождения векторного произведения векторов предназначена специальная функция *cross*, например:

```
» u=[1 2 3]; v=[3 2 1];
» cross(u,v)
ans =
   -4     8    -4
```

Скалярное произведение векторов можно вычислить с помощью функции общего назначения *sum*, вычисляющей сумму всех элементов векторов (для матриц эта функция вычисляет суммы для всех столбцов). Скалярное произведение, как известно, равно сумме произведений соответствующих координат (элементов) векторов. Таким образом, выражение:

```
» sum(u.*v)
```

вычисляет скалярное произведение двух векторов u и v. Скалярное произведение можно также вычислить как: u*v′.

Длина вектора вычисляется с помощью скалярного произведения и функции извлечения квадратного корня, например:

```
» sqrt(sum(u.*u))
```

Ранее рассмотренные для скаляров операции отношения и логические операции выполняются в случае массивов поэлементно. Оба операнда должны быть одинаковых размеров, при этом операция возвращает результат такого же размера. В случае, когда один из операндов скаляр, производится его предварительное расширение, смысл которого уже был пояснен на примере арифметических операций.

Среди функций, генерирующих матрицы с заданными свойствами, упомянем здесь функцию *eye*, производящую единичные квадратные матрицы, а также широко применяемую на практике функцию *rand*, генерирующую массив со случайными элементами, равномерно распределенными на интервале от 0 до1. Например, выражение

```
» F=rand(3)
```

порождает массив случайных чисел размером 3x3 с элементами, равномерно распределенными на интервале от 0 до 1.

Если вызвать эту функцию с двумя аргументами, например *R=rand(2,3),*то получится матрица *R* случайных элементов размером 2x3. При вызове функции *rand* с тремя и более скалярными аргументами производятся многомерные массивы случайных чисел.

Определитель квадратной матрицы вычисляется с помощью функции *det.*

Среди функций, производящих простейшие вычисления над массивами, помимо рассмотренной выше функции *sum*, упомянем еще функцию *prod*, которая во всем аналогична функции *sum*, только вычисляет она не сумму элементов, а их произведение. Функции *max* и *min* ищут соответственно максимальный и минимальный элементы массивов. Для векторов они возвращают единственное числовое значение, а для матриц они порождают набор экстремальных элементов, вычисленных для каждого столбца. Функция *sort* сортирует в возрастающем порядке элементы одномерных массивов, а для матриц она производит такую сортировку для каждого столбца отдельно.

Наконец, рассмотрим уникальную возможность М-языка системы MATLAB производить групповые вычисления над массивами, используя обычные математические функции, которые в традиционных языках программирования работают только со скалярными аргументами. В результате с помощью крайне компактных записей, удобных для ввода с клавиатуры в интерактивном режиме работы с командным окном системы MATLAB, удается произвести большой объем вычислений. Например, всего два коротких выражения

```
» x=0:0.01:pi/2; y=sin(x);
```

вычисляют значения функции *sin* сразу в 158 точках, формируя два вектора *x* и *y* со 158 элементами каждый.

Построение графиков функци

Графические возможности системы MATLAB являются мощными и разнообразными. Изучим наиболее простые в использовании возможности (высокоуровневую графику).

Сформируйте два вектора *x* и *y:*

```
» x=0:0.01:2; y=sin(x);
```

Вызовите функцию:

```
» plot(x,y)
```

и вы получите на экране график функции (рис. 1).

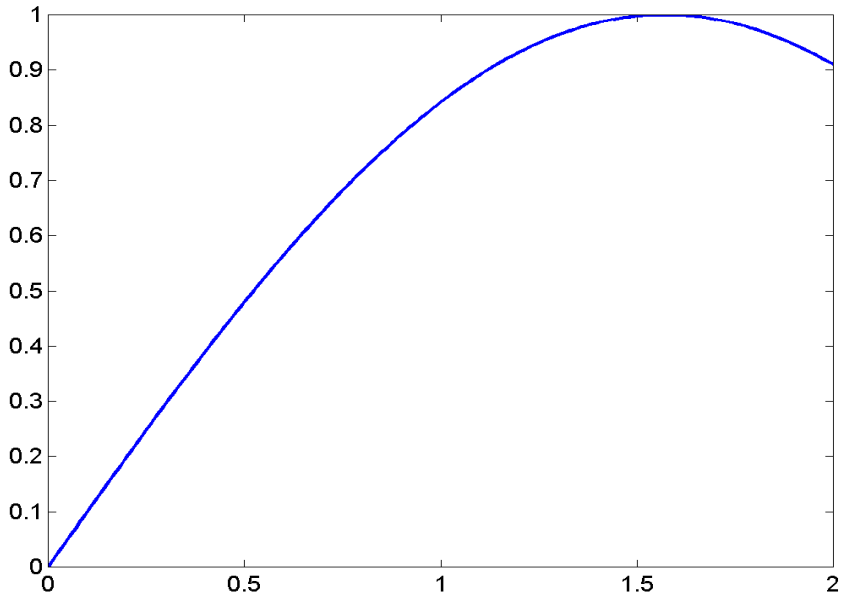

Рис. 1. График функции y=sin(x)

MATLAB показывает графические объекты в специальных графических окнах, имеющих в заголовке слово *Figure.*

Не убирая с экрана дисплея первое графическое окно, введите с клавиатуры выражения

```
» z=cos(x);
» plot(x,z)
```

и получите новый график функции в том же самом графическом окне (при этом старые оси координат и график пропадают – этого также можно добиться командой *clf*, командой *cla* удаляют только график с приведением осей координат к их стандартным диапазонам от 0 до 1).

Если нужно второй график провести «поверх первого графика», то перед вторичным вызовом графической функции *plot* нужно выполнить команду *hold on,* которая предназначена для удержания текущего графического окна:

```
» x=0:0.01:2; y=sin(x);
» plot(x,y)
```

```
» z=cos(x);
» hold on
» plot(x,z)
```

Практически тоже самое получится (рис. 2), если набрать:

```
» x=0:0.01:2; y=sin(x); z=cos(x);
» plot(x,y,x,z)
```

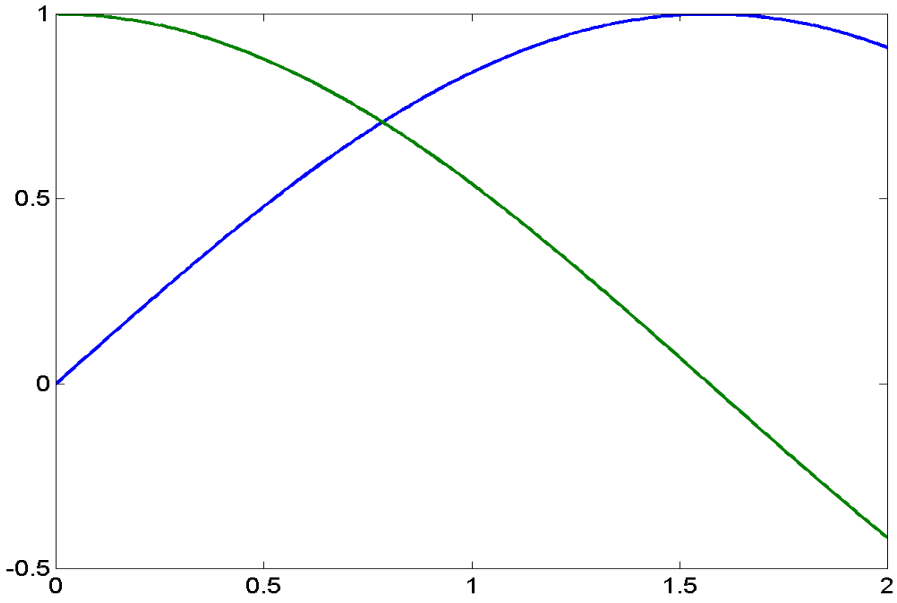

Рис. 2. Графики функций y=sin(x), z=cos(x), построенные в одном графическом окне

Если нужно одновременно визуализировать несколько графиков так, чтобы они не мешали друг другу, то это можно сделать двумя способами. Первым решением является построение их в разных графических окнах. Для этого перед вторичным вызовом функции *plot* следует набрать команду *figure*, которая создает новое графическое окно и заставляет все последующие за ней функции построения графиков выводить их туда.

Вторым решением показа нескольких графиков без конфликта диапазонов осей координат является использование функции *subplot.* Эта функция позволя-

ет разбить область вывода графической информации на несколько подобластей, в каждую из которых можно вывести графики различных функций.

Например, для ранее выполненных вычислений с функциями *sin и cos* постройте графики этих двух функций в первой подобласти, а график функции *exp(x)* – во второй подобласти одного и того же графического окна (рис. 3):

```
» w=exp(x);
» subplot(1,2,1); plot(x,y,x,z)
» subplot(1,2,2); plot(x,w)
```

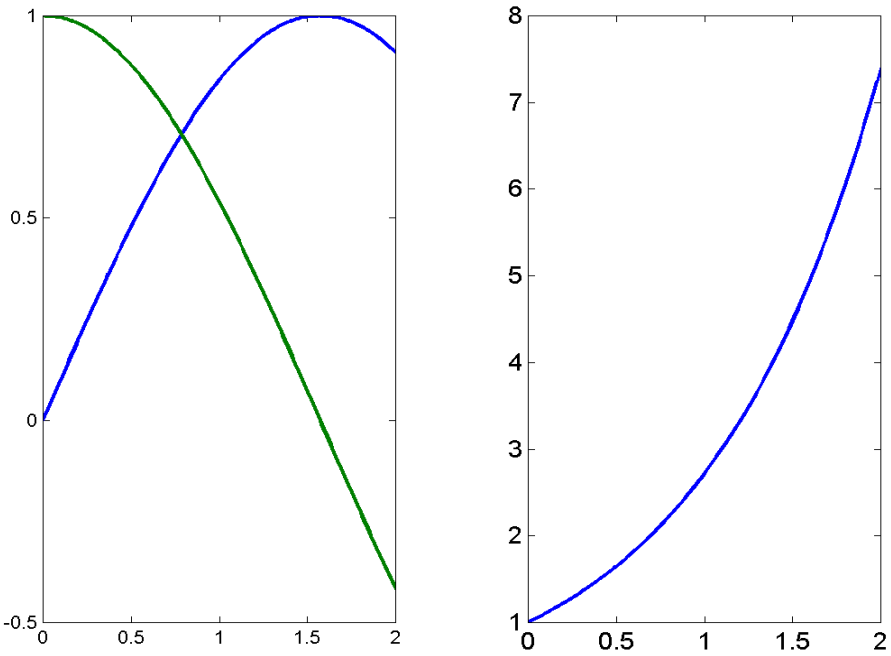

Рис. 3. Графики функций y=sin(x), z=cos(x) и w=exp(x), построенные в двух подобластях одного графического окна

Диапазоны изменения переменных на осях координат этих подобластей независимы друг от друга. Функция *subplot* принимает три числовых аргумента, первый из которых равен числу рядов подобластей, второй равен числу колонок подобластей, а третий аргумент – номеру подобласти (номер отсчитывается

вдоль рядов с переходом на новый ряд по исчерпании). Снять действие функции *subplot* можно командой:

```
» subplot(1,1,1)
```

Если для одиночного графика диапазоны изменения переменных вдоль одной или обеих осей координат слишком велик, то можно воспользоваться функциями построения графиков в *логарифмических масштабах.* Для этого предназначены функции *semilogx, semilogy* и *loglog.*

Построить график функции в полярных координатах (рис. 4) можно с помощью графической функции *polar.*

```
» phi=0:0.01:2*pi; r=sin(3*phi);
» polar(phi,r)
```

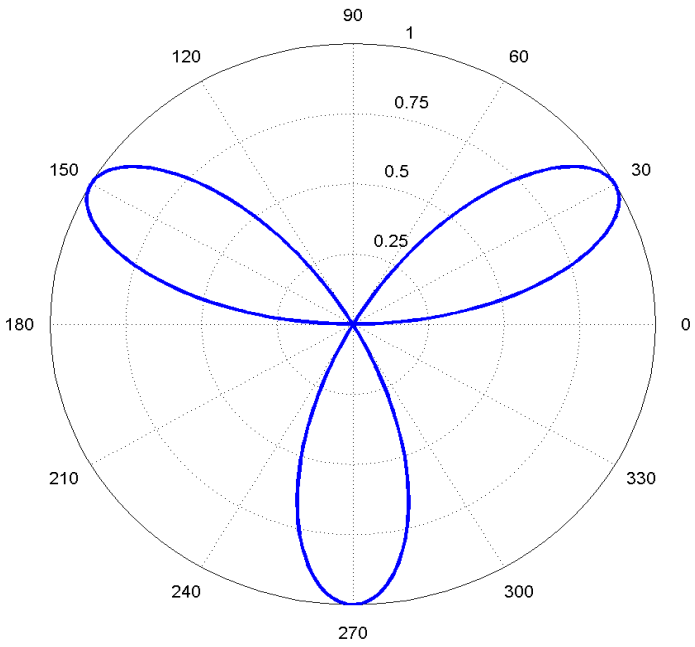

Рис. 4. График функции r=sin(3*phi) в полярных координатах

Рассмотрим дополнительные возможности, связанные с управлением внешним видом графиков – задание цвета и стиля линий, а также размещение различных надписей в пределах графического окна. Например, команды

```
» x=0:0.1:3; y=sin(x);
» plot(x,y,'r-',x,y,'ko')
```

позволяют придать графику вид красной сплошной линии (рис. 5), на которой в дискретных вычисляемых точках проставляют черные окружности. Здесь функция *plot* дважды строит график одной и той же функции, но в двух разных стилях. Первый из этих стилей отмечен как *'r-'*, что означает проведение линии красным цветом (буква *r*), а штрих означает проведение сплошной линии. Второй стиль, помечен как *'ko'*, означает проведение черным цветом (буква *k*) окружностей (буква *o*) на месте вычисляемых точек.

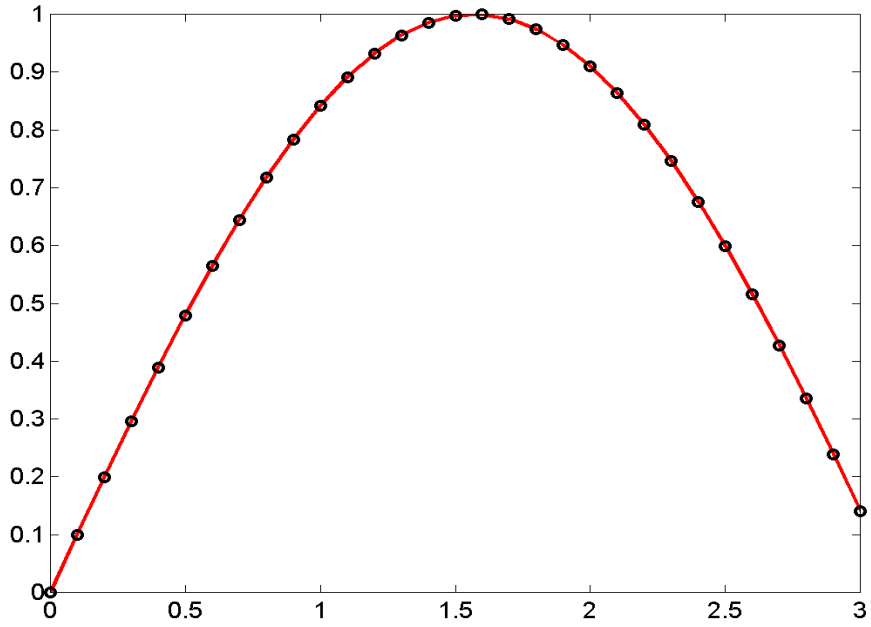

Рис. 5. Построение графика функции y=sin(x) в двух разных стилях

В общем случае функция *plot (x1, y1, s1, x2, y2, s2, ...)* позволяет объединить в одном графическом окне несколько графиков функций *y1(x1), y2(x2), ...* проведя их со *стилями s1, s2, ...* и т. д.

Стили *s1, s2,...* задаются в виде набора трех *символьных маркеров,* заключенных в одиночные кавычки (апострофы). Один из этих маркеров задает тип линии (Таблица 3). Другой маркер задает цвет (Таблица 4). Последний маркер задает тип проставляемых «точек» (Таблица 5). Можно указывать не все три маркера. Тогда используются маркеры, установленные по умолчанию. Порядок,

в котором указывают маркеры, не является существенным, то есть *'r+-'* и *'-+r'* приводит к одинаковому результату.

Таблица 3.

Маркеры, задающие тип линии

Маркер	-	--	:	-.
Тип линии	Непрерывная	Штриховая	Пунктирная	Штрихпунктирная

Таблица 4

Маркеры, задающие цвет линии

Маркер	Цвет линии	Маркер	Цвет линии
c	Голубой	g	Зеленый
m	Фиолетовый	b	Синий
y	Желтый	w	Белый
r	Красный	k	Черный

Таблица 5

Маркеры, задающие тип точки

Маркер	.	+	*	o	x
Тип точки	Точка	Плюс	Звездочка	Кружок	Крестик

Если в строке стиля поставить маркер на тип точки, но не проставить маркер на тип линии, то тогда отображаются только вычисляемые точки, а непрерывной линией они не соединяются.

Теперь перейдем к оформлению осей координат, к надписям на осях. Система MATLAB устанавливает пределы на горизонтальной оси равными тем значениям, что указаны пользователем для независимой переменной. Для зависимой переменной по вертикальной оси MATLAB самостоятельно вычисляет диапазон изменения значений функции. Если мы хотим отказаться от этой осо-

бенности масштабирования при построении графиков в системе MATLAB, то мы должны явным образом навязать свои пределы изменения переменных по осям координат. Это делается с помощью функции

axis ([xmin, xmax, ymin, ymax])

Для проставления различных надписей на полученном рисунке применяют функции *xlabel, ylabel, title* и *text.* Функция *xlabel* создает подпись у горизонтальной оси, функция *ylabel* – тоже для вертикальной оси (причем эти надписи ориентированы вдоль осей координат). Если требуется разместить надпись в произвольном месте рисунка, применяем функцию *text*. Общий заголовок для графика создается функцией *title.* Кроме того, используя команду *grid on,* можно нанести измерительную сетку на всю область построения графика. Например (рис. 6):

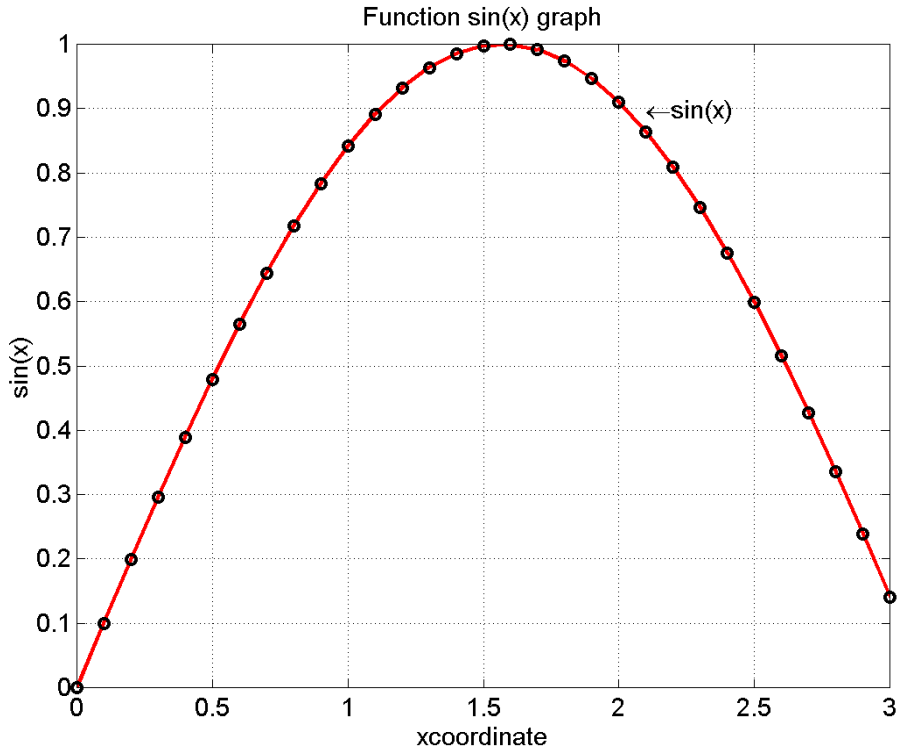

Рис. 6. График функции y=sin(x), построенный в двух стилях, с надписями на координатных осях и на рисунке

```
» x=0:0.1:3; y=sin(x);
» plot(x,y,'r-',x,y,'ko')
» title('Function sin(x) graph');
» xlabel('xcoordinate'); ylabel('sin(x)');
» text(2.1, 0.9, '\leftarrowsin(x)'); grid on
```

Надпись функцией *text* помещается начиная от точки с координатами, указанными первыми двумя аргументами. По умолчанию координаты задаются в тех же единицах измерения, что и координаты, указанные на горизонтальной и вертикальной осях. Специальные *управляющие символы* вводятся внутри текста после символа \ (обратная косая черта).

Трехмерная графика

Возможности отображения трехмерных графических объектов в системе MATLAB весьма обширны. Мы сосредоточимся на изображении пространственных линий и на построении графиков функций двух вещественных переменных, которые представляют поверхности в пространстве.

Каждая точка в пространстве характеризуется тремя координатами. Набор точек, принадлежащих некоторой линии в пространстве, нужно задать в виде трех векторов, первый из которых содержит первые координаты этих точек, второй вектор – вторые их координаты, ну а третий вектор - третьи координаты. После чего эти три вектора можно подать на вход функции *plot3*, которая и осуществит проектирование соответствующей трехмерной линии на плоскость и построит результирующее изображение (рис. 7).

Введите с клавиатуры:

```
» t=0:pi/50:10*pi;
» x=sin(t);
» y=cos(t); plot3(x,y,t); grid on
```

Убедитесь, что получилась винтовая линия.

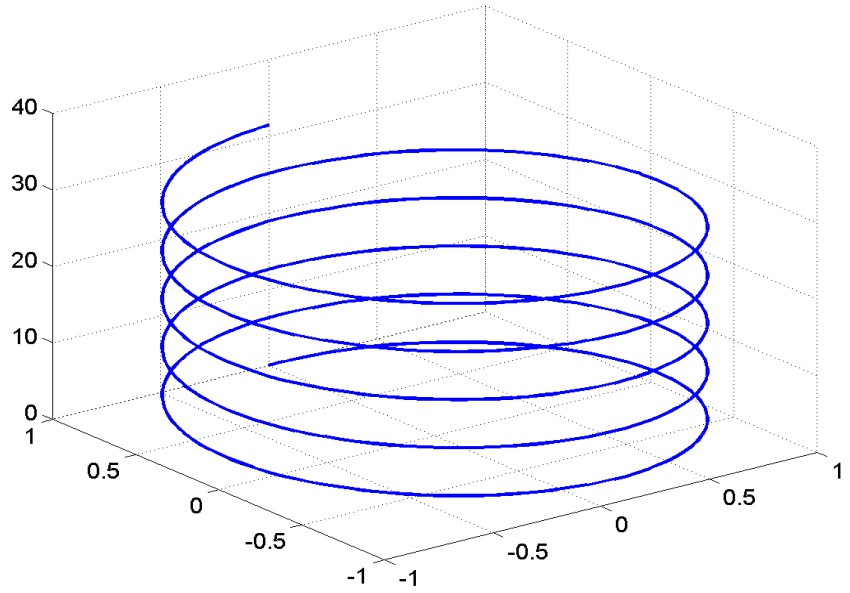

Рис. 7. График винтовой линии, построенный с помощью функции *plot3*

Эту же функцию *plot3* можно применить и для изображения поверхностей в пространстве, если, конечно, провести не одну линию, а много. Наберите с клавиатуры:

```
» u=-2:0.1:2; v=-1:0.1:1;
» [X,Y]=meshgrid(u,v);
» z=exp(-X.^2-Y.^2);
» plot3(X,Y,z)
```

Получите трехмерное изображение графика функции (рис. 8).

Функция *plot3* строит график в виде набора линий в пространстве, каждая из которых является сечением трехмерной поверхности плоскостями, параллельными плоскости *yOz*. Помимо этой простейшей функции система MATLAB располагает еще рядом функций, позволяющих добиваться большей реалистичности в изображении трехмерных графиков.

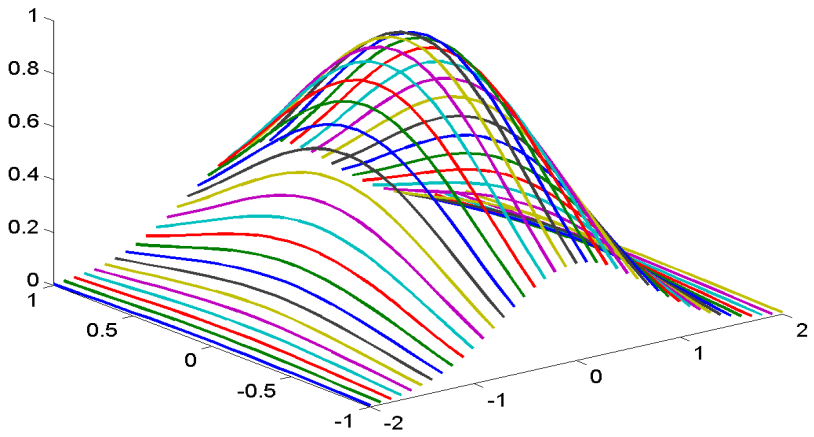

Рис. 8. График поверхности в пространстве, построенный с помощью функции *plot3*

Сценарии и m-файлы.

Для простых операций удобен интерактивный режим, но если вычисления нужно многократно выполнять или необходимо реализовывать сложные алгоритмы, то следует использовать m-файлы MATLAB (расширение файла состоит из одной буквы m). Познакомимся со *script-m-файлами* (или сценариями) – текстовыми файлами, содержащими инструкции на языке MATLAB, подлежащими исполнению в автоматическом пакетном режиме. Создать такой файл удобнее с помощью редактора системы MATLAB. Он вызывается из командного окна системы MATLAB командой меню *File/New/M-file* (или самой левой кнопкой на полосе инструментов, на которой изображен чистый белый лист бумаги). Записанные в script-файлы команды будут выполнены, если в командной строке ввести имя script-файла (без расширения). Переменные, определяемые в командном окне и переменные, определяемые в сценариях, составляют единое рабочее пространство системы MATLAB, причем переменные, определяемые в сценариях, являются глобальными, их значения заместят значения та-

ких же переменных, которые были использованы до вызова данного *script-файла*.

После создания текста сценария его надо сохранить на диске в удобном для вас каталоге. Путь к этому каталогу обязательно должен быть известен системе MATLAB. Командой *File/Set Path* вызывается диалоговое окно просмотрщика путей доступа к каталогам. Для добавления нового каталога в список путей доступа необходимо выполнить далее команду меню *Path/Add to path*.

Порядок выполнения лабораторной работы 1.

Задание 1. Задать матрицу A с помощью операции конкатенации:

$$\begin{pmatrix} 3,25 & -1,07 & 2,34 \\ 10,10 & 0,25 & -4,78 \\ 5,04 & -7,79 & 3,31 \end{pmatrix}.$$

Задание 2. Сгенерируйте массив B размером 3x3 со случайными элементами, равномерно распределенными на интервале от 0 до 1.

Задание 3. Выполните действия:

$A+10\cdot B,\quad A\cdot B,\quad B^T$,

почленно умножить A на B,

расположить элементы матрицы A по возрастанию (по столбцам),

определить максимальный и минимальный элементы матрицы B,

вычислить определитель матрицы B.

Задание 4. Задать массив C, используя операцию индексации и одну из функций: ones или zeros:

$$\begin{pmatrix} 0 \\ 5.71 \\ -3.61 \end{pmatrix}.$$

Задание 5. Решить систему алгебраических линейных уравнений:

$$A\cdot X = C\,.$$

Задание 6. Определить массив D:

$$D = \left|\sin(A) + B^{3/5}\right|.$$

Задание 7. Для двух векторов:

$$\vec{a} = \{3,2;\quad 2,8;\quad -1,4\} \text{ и } \vec{b} = \{0,6;\quad 3,2;\quad -4,8\}$$

определите: $\vec{a}\cdot\vec{b};\quad \vec{a}\times\vec{b};\quad \left|\vec{a}\right|$.

Задание 8. Постройте два графика в рамках одних осей координат:

$$y = e^{-x^2}$$
$$z = \text{arctg}(x^{1/2})$$
$$x \in [0, 4\pi]$$

Сделайте надписи на осях, заголовок для графика, пояснительную надпись на рисунке. Задайте самостоятельно тип линий и цвет.

Задание 9. Построить графики функций y(x) и z(x) в разных подобластях одного графического окна. Интервалы изменения для x определите самостоятельно.

Задание 10. Постройте поверхность:

$$f(x, y) = \ln(x^2 + y^2 - xy)$$
$$x, y \in [1, \quad 2]$$

Контрольные вопросы

1. Как изменить на экране формат вывода числа?
2. Как можно просмотреть в MATLAB список всех элементарных математических функций?
3. Какие виды функций в MATLAB Вам известны?
4. Опишите способы создания одномерных массивов в MATLAB.
5. Опишите способы создания двумерных массивов в MATLAB.
6. Перечислите и объясните действие операторов, используемых при вычислениях с массивами.
7. Опишите действие операций отношения.
8. Опишите действие логических операций.
9. Как построить декартовый и полярный графики функции одной переменной?
10. Как построить несколько графиков в одной системе координат?
11. Как построить графики в разных подобластях одного графического окна?

12. Как изменить цвет и стиль линий на графиках?
13. Как сделать надписи на осях, на полученном рисунке? Как сделать заголовок для графика?
14. Как построить график функции двух переменных? Как построить график поверхности?
15. Что такое m-файлы? Как создать, сохранить и вызвать m-файл?

Лабораторная работа 2

Решение типовых задач алгебры и анализа

Решение систем линейных уравнений

В системе MATLAB для решения систем линейных уравнений предусмотрены знаки операций / и \. Чтобы решить систему линейных уравнений вида

$$A \cdot y = B,$$

где A – заданная квадратная матрица размером *N*x*N*, а B – заданный вектор – столбец длины *N*, достаточно применить операцию \ и вычислить выражение $A \backslash B$.

Операция \ называется *левым делением матриц* и, будучи примененная к матрицам A и B в виде $A \backslash B$, примерно эквивалентна вычислению выражения $inv(A) * B$. Здесь под $inv(A)$ понимается вычисление матрицы, обратной к матрице A.

Операцию / называют *правым делением матрицы*. Выражение A / B примерно соответствует вычислению выражения $B * inv(A)$. Значит, эта операция позволяет решать системы линейных уравнений вида

$$y \cdot A = B.$$

Нахождение нулей функций

Решение уравнения $F(x) = 0$, или нахождение нулей функции, осуществляется с помощью функции:

$fzero(name, \quad x0)$.

В качестве первого аргумента ей передается имя функции, задающей исходное уравнение, вторым аргументом служит начальное приближение к корню. Возвращаемым значением функции $fzero$ является нуль функции $name$ в окрестности точки $x0$. Определим, в качестве примера, нули функции $\cos(x)$ на отрезке от 0 до *pi*. В качестве начального приближения примем $x0 = 1$.

```
» x=fzero('cos', 1)
x = 1.5708
```

Если требуется найти корень функции, отличной от стандартной (встроенной в систему MATLAB) и тем самым не имеющей в рамках системы MATLAB фиксированного имени, то нужно приписать некоторое имя выражению, вычисляющему функцию.

Пусть, например, требуется найти корни уравнения $\cos(x) = x$, что эквивалентно нахождению нулей функции, вычисляемой по формуле $y = \cos(x) - x$, не имеющей в рамках системы MATLAB фиксированного имени. В этом случае нужно в любом простейшем текстовом редакторе (можно в окне редактора-отладчика MATLAB) набрать две строки следующего кода:

```
function y=MyFunction1(x)
y=cos(x)-x;
```

и запомнить их в файле *MyFunction1.m*, который нужно разместить в текущем каталоге системы MATLAB (узнать его можно командой *cd*). После этого можно воспользоваться функцией *fzero:*

```
» x=fzero('MyFunction1',pi/2)
x = 0.7391
```

Если найдено абсолютно точное значение корня, то значение функции в этой точке равно нулю. Таким образом, величина функции в приближенно найденном нуле косвенно характеризует погрешность результата. Чтобы управляв погрешностью, нужно осуществлять вызов функции *fzero* с тремя аргументами:

```
fzero( name, x0, tol ),
```

где параметр *tol* задает требуемую величину погрешности (ошибки). Повторив предыдущие вычисления, потребовав большей точности расчетов (то есть меньшей погрешности):

```
»format long
x = fzero('MyFunction1',pi/2, le-8)
```

x =0.73908513263090

MyFunction1(x)

ans = 9.778322596076805e-010,

откуда видно, что действительно достигнута большая точность нахождения нуля функции.

Еще раз подчеркнем, что функция *fzero* находит нули только вещественнозначных функций одной вещественной переменной. Однако часто бывает необходимо найти комплексные корни вещественнозначных функций, особенно в случае многочленов. Для этой цели в системе MATLAB существует специальная функция *roots*, которой в качестве аргумента передается массив коэффициентов многочлена. Например, для многочлена $x^4 - 3x^3 + 3x^2 - 3x + 2$, имеющего два вещественных (1 и 2) и два комплексных корня (i и -i), нужно сначала сформировать массив его коэффициентов (расположив по порядку убывания степени x):

Coef = [1, -3, 3, -3, 2],

после чего вызвать функцию *roots:*

r = roots(Coef)

r = 2.00000000000000

0.00000000000000 +1.OOOOOOOOOOOOOOi

0.00000000000000 -1.OOOOOOOOOOOOOOi

1.00000000000000

В задаче о нахождении нулей функции сложным моментом является нахождение начального приближения к нулю функции, а также априорная оценка их количества. Поэтому важно параллельно с применением функций типа *roots* или *fzero* визуализировать поведение искомых функций на том или ином отрезке значений аргумента (построить график функции).

Поиск минимума функции

В системе MATLAB имеются специальные функции для поиска минимума заданных функций. При этом возможен поиск минимума как для функции од-

ной вещественной переменной, так и для функций многих переменных.

Для функций одной переменной их минимумы разыскивает функция *fmin*:

```
fmin( name, x0, xl ).
```

Здесь *name* представляет имя функции, у которой находятся минимумы, а *x0* и *xl* задают отрезок поиска.

Для поиска минимума функции нескольких переменных применяется функция *fmins:*

```
xmin = fmins( name, x0 ).
```

Здесь *name* является именем функции нескольких переменных, для которой ищется минимум, а *x0* - это вектор ее аргументов, с которого начинается поиск. Для иллюстративного примера создадим простую функцию двух переменных имеющую минимумом точку (0,0).

```
function y = MyFunc2 ( x )
y = x(1)^2+ x(2)^2;
```

Этот текст надо записать в файл *MyFunc2.m* в текущий каталог системы MATLAB. После этого можно вызвать функцию fmins:

```
xmin = fmins( 'MyFunc2', [1,1] );
```

которая приближенно находит вектор *xmin* координат точки минимума:

```
xmin(1)
ans =-2.102352926236483e-005
xmin(2)
ans =2.548456493279544e-005
```

Обе найденные координаты близки к своим точным значениям, равным нулю.

Для функций нескольких переменных еще важнее, чем для ранее рассмотренных функций одной вещественной переменной, постараться априорно оценить количество и приблизительное нахождение локальных минимумов, и тут могут существенно помочь трехмерные графики.

Вычисление определенных интегралов

Для вычисления интегралов методом трапеций в системе MATLAB предусмотрена функция *trapz:*

```
Integ = trapz ( x, y ) ;
```

Одномерный массив *x* (вектор) содержит дискретные значения аргументов подынтегральной функции. Значения подынтегральной функции в этих точках сосредоточены в одномерном массиве *y*. Чаще всего для интегрирования выбирают равномерную сетку, то есть значения элементов массива *x* отстоят друг от друга на одну и ту же величин (шаг интегрирования). Точность вычисления интеграла зависит от величины шага интегрирования: чем меньше этот шаг, тем больше точность.

Вычислим простой интеграл методом трапеций с шагом интегрирования *pi/10*.

$$\int_0^{\pi} \cos(x)dx .$$

```
dx = pi/10;
x = 0:dx:pi;
y=cos(x);
I1 = trapz(x,y);
I1 = 5.5511e-017
```

Обычно для достижения высокой точности требуется выполнять интегрирование с очень малыми шагами, а контроль достигнутой точности осуществлять путем сравнения последовательных результатов. При одном и том же шаге интегрирования методы более высоких порядков точности достигают более точных результатов.

В системе MATLAB методы интегрирования более высоких порядков точности реализуются функциями: *quad* (метод Симпсона) и *quad8* (метод Ньютона-Котеса 8-го порядка точности). Оба этих метода являются к тому же *адап-*

тивными. Последнее означает, что пользователю нет необходимости контролировать достигнутую точность результата путем сравнения последовательных значений, соответствующих разным шагам интегрирования. Все это указанные адаптивные функции выполняют самостоятельно.

У функции *quad8* более высокий порядок точности по сравнению с функцией *quad*, что очень хорошо для гладких функций, так как обеспечивается более высокая точность результата при большем шаге интегрирования (меньшем объеме вычислений). Как и многие другие функции системы MATLAB, функции *quad* и *quad8* могут принимать различное количество параметров. Минимальный формат вызова этих функций

```
quad8(name, x1, x2)
```

включает в себя три параметра: имя подынтегральной функции – *name*, нижний предел интегрирования – *x1* и верхний предел интегрирования – *x2*. Если применяется четвертый параметр, то он является требуемой относительной точностью результата вычислений. Кстати, если обе эти адаптивные функции не могут обеспечить получение необходимой точности (расходящийся или близкий к этому интеграл), то они возвращают символическую бесконечность *Inf*.

Из высшей математики известно, что к определенным интегралам могут быть сведены многие другие типы интегралов, например криволинейные интегралы. Таким образом, с помощью функций *quad, quad8* (или *trapz*) можно вычислить и эти интегралы.

Рассмотрим пример на криволинейные интегралы первого рода. Пусть требуется вычислить массу *M* винтовой линии *C*:

$$x = sin(t);\ y = 2cos(t);\ z = 3t;\ 0 <= t <= 2$$

с постоянной линейной плотностью, равной 5. Задача решается с помощью *криволинейного интеграла первого рода:*

$$M = \int_C 5ds\ ,$$

который сводится к вычислению следующего обыкновенного определенного интеграла:

$$M = 5\int_0^2 \sqrt{(x')^2 + (y')^2 + (z')^2}\,dt .$$

Для вычисления подынтегральной функции создадим следующий текст:

```
function z = MyFunc321( t )
z = sqrt( cos(t).^2 + 4*sin(t).^2 + 9 ),
```

который запишем в файл *MyFunc321.m*, после чего вызываем функцию *quad:*

```
M = 5 * quad( 'MyFunc321', 0, 2 );
M = 34.2903
```

Двойные интегралы в системе MATLAB вычисляются с помощью специальной функции *dblquad.* Вычислим интеграл:

$$\int_0^1 \int_1^2 (x\sin(y) + y\sin(x))dxdy .$$

Оформим подынтегральную функцию в следующем виде:

```
function z = Fof2Var( x, y )
z = x.*sin(y) + y.*sin(x);
```

(записав этот текст в файл *Fof2Var .m*) и вызовем функцию *dblquad:*

```
J = dblquad( 'Fof2Var', 0, 1, 1, 2 );
J = 1.1678
```

Решение систем обыкновенных дифференциальных уравнений

Для решения систем обыкновенных дифференциальных уравнений в системе MATLAB имеются функции: *ode23, ode45, odell3, odel5s, ode23s, ode23t* и *ode23tb.*

Функции с суффиксом s предназначены для решения так называемых *систем жестких дифференциальных уравнений*, а для всех остальных систем дифференциальных уравнений наиболее употребительной является функция

ode45, реализующая алгоритм Рунге- Кутта 4–5-го порядка (разные порядки точности используются для контроля шага интегрирования).

Пусть необходимо решить систему *n* дифференциальных уравнений, разрешенных относительно первых производных функций *y1, y2, ..., yn* (это все функции от *x*):

$$y1' = Fl(x, y1, y2, ..., yn);$$
$$y2' = F2(x, y1, y2, ..., yn);$$
$$...$$
$$yn' = Fn(x, y1, y2, ..., yn).$$

Введем вектор-столбцы *Y* и *F*, состоящие из *y1, y2, ..., yn* и *Fl, F2, ..., Fn*, соответственно. Тогда система дифференциальных уравнений примет следующий векторный вид:

$$Y' = F(x, Y).$$

Чтобы применить «решатель» *ode45*, нужно оформить в виде функции пользователя правую часть системы уравнений *F(x, Y)*.

Пусть, к примеру, требуется решить дифференциальное уравнение:

$$y'' + y + K \cdot x^2 = 0$$

с начальными условиями:

$$y(0) = 1$$
$$y'(0) = 0.$$

Данное дифференциальное уравнение второго порядка приведем к системе дифференциальных уравнений первого порядка:

$$y1' = y2 + K * x * x;$$
$$y2' = -y1.$$

с начальными условиями *y1(0) = 0, y2(0) = 1*. Здесь *K*- коэффициент нелинейности задачи. При *K* = 0 задача становится чисто линейной и описывает гармонические колебания. Если постепенно увеличивать значение коэффициента *K* и находить соответствующие решения, то можно будет наглядно наблюдать постепенное проявление нелинейного характера колебаний.

Для данного примера неизвестная вектор-функция *Y* состоит из двух элементов:

Y = [y1, y2].

Так как функции *y1* и *y2* вычисляются во многих точках в процессе нахождения решения, то реально *y1* и *y2* являются вектор-столбцами. Вектор *F* правых частей системы уравнений для, *K* = 0.01, вычисляем с помощью собственной функции *MyDifEql:*

```
function F = MyDifEql( x, y)
F = [ 0.01 * x * x + y(2); -y(l) ];
```

текст которой записываем в файл *MyDifEql.m*. Эта функция вызывается каждый раз, когда требуется вычислить правые части уравнений в конкретной точке *x*, так что здесь *x* является скаляром, а вектор *y* состоит всего из двух элементов. Теперь можно вызывать функцию *ode45,*находящую решение нашей системы дифференциальных уравнений с начальными условиями [0,1] на отрезке [0,20].

```
[ X, Y ] = ode45( 'MyDifEql',[0,20],[0,1] );
```

Решения уравнений с помощью команды:

```
plot (X, Y(:, 1), X, Y(:, 2))
```

отображаются на графике (Y(:, 1) – первый столбец матрицы решений, Y(:, 2) – второй ее столбец).

Теперь перейдем к так называемым *жестким системам дифференциальных уравнений*, решения которых на разных отрезках изменения независимых переменных ведут себя абсолютно по-разному: в одних местах наблюдается чрезвычайно быстрое изменение зависимых переменных, в то время как в других местах имеет место их сверхмедленная эволюция. Это слишком сложный характер поведения для обычных алгоритмов численного решения дифференциальных уравнений. Поэтому для надежного решения жестких систем уравнений применяют специальные методы. Решим дифференциальное уравнение Ван-дер-Поля, которое описывает *нелинейные релаксационные колебания* в различных электронных устройствах:

$$y1' = y2;$$

$$y2' = -y1 + K * (1 - y1 * y1) * y2$$

с начальными условиями *y1(0) =2, y2(0) = 0.* Здесь K = 1000 - большой коэффициент нелинейности задачи. Составим функцию *MyVanDerPol,* которая описывает правые части представленных дифференциальных уравнений:

```
function F = MyVanDerPol( x, y )
F = [ y(2); -y(1) + 1000*( 1 - y(1)^2 ) * y(2) ].
```

Для решения дифференциальных уравнений Ван-дер-Поля на отрезке изменения независимой переменной [0, 3000] используем «решатель» *odel5s:*

```
[X, Y] = odel5s ( 'MyVanDerPol', [0,3000], [2,0] ),
```

после чего визуализируем решение.

Символьные вычисления.

Возможности встроенного пакета символьных вычислений и операций *Symbolic Math Toolbox* достаточно обширны, рассмотрим лишь некоторые его возможности.

Под символьным объектом в системе MATLAB понимается переменная, предназначенная для символьных преобразований. Объявление такой переменной осуществляется функцией *sym* либо *syms.* Функция *sym* используется при объявлении какой-либо одной переменной символьной, например:

sym x – объявление символьной переменной x.

Упростим выражение: (используется функция *simplify*)

$$\frac{x^2 - y^2}{x - y}.$$

```
» sym x;
» sym y;
» simplify((x^2-y^2)/(x-y))
ans = x+y
```

Функция *syms* позволяет объявлять сразу несколько символьных переменных, которые необходимо отделять друг от друга пробелами, например:

```
» syms x y z;
```

Упрощение тригонометрических, логарифмических, экспоненциальных функций и полиномов осуществляется функцией *expand*, формат обращения к которой имеет следующий вид:

```
rez=expand(S) ,
```

где S – символьное выражение, которое нужно упростить;

rez – результат упрощения. Например:

```
» syms x y;
» rezl=expand(sin(x+y) }
rezl = sin (x) *cos (y) +cos (x) *sin (y)
```

Для выполнения операции дифференцирования используется функция *diff*, формат обращения к которой имеет следующий вид:

```
diff(y(x)),
```

где y(x) - явно заданная функция. Ниже приведен пример, иллюстрирующий работу этой функции:

```
» syms x;
» D=diff (x^2+4*x^5)
D = 2*x+20*x^4
```

Для того чтобы продифференцировать заданную функцию n раз, нужно обратиться к ней следующим образом:

```
D =diff (S, 'v' ,n),
```

где S – дифференцируемое выражение, v – переменная дифференцирования.

Следующий пример иллюстрирует дифференцирование заданного выражения S дважды по переменной x.

```
» syms x y;
» S=x^3*y^2+sin(x*y) ;
» D=diff (S,'x' ,2)
D = 6*x*y^2-sin (x*y)*y^2
```

Для решения дифференциальных уравнений в MATLAB зарезервирована функция *dsolve*, которая имеет следующие форматы обращения:

1. y=dsolve('Dy(x)') ,

где Dy(x) -уравнение; y - возвращаемые функцией *dsolve* решения.

2. y=dsolve ('Dy (x)' , ' НУ ') ,

где Dy(x) -уравнение; НУ - начальные условия. Первая производная функции обозначается Dy, вторая производная – D2y и так далее.

Функция *dsolve* предназначена также для решения системы дифференциальных уравнений. В этом случае она имеет следующий формат обращения:

[f,g]=dsolve('Df(x),Dg(x)', 'НУ'),

где Df(x) ,Dg(x) - система уравнений; НУ - начальные условия.

Например: решить дифференциальное уравнение:

$$y'(x) = 0.6-0.2y(x)$$

с начальным условием y(0)=0.

» dsolve(' Dy = 0.6 - 0.2*y ', ' y(0)=0 ')

ans = 3+exp(-l/5*t)*3.

Перечислим еще несколько функций, часто используемых при символьных вычислениях:

expand – раскрывает алгебраические и функциональные выражения;

factor – раскладывает многочлены на простейшие множители;

det – вычисляет определитель символьной матрицы;

inv – вычисляет обратную матрицу;

int – вычисляет неопределенный интеграл;

limit –вычисляет пределы;

taylor – осуществляет разложение функций в ряд Тейлора;

solve – решает алгебраическое уравнение и систему алгебраических уравнений.

Порядок выполнения лабораторной работы 2.

Задание 1. Решить систему уравнений:

$$\begin{aligned} x + y - z &= 36 \\ x + z - y &= 13 \\ y + z - x &= 7 \end{aligned}.$$

Задание 2. Определить абсциссы точек пересечения графиков функций:

$$y = -3 \cdot x^3, \quad y = x^4 - 2 \cdot x^2 + 10.$$

Задание 3. Вычислить предел:

$$\lim_{x \to 0} \left(\cos x\right)^{\frac{1}{x \sin x}}.$$

Задание 4. Найти производную от функции y(x):

$$\begin{aligned} &a) \quad y = \ln \sqrt{\exp(2x) + 1} \\ &b) \quad x^2 y^2 + 2\ln(xy) = 4 \end{aligned}.$$

Задание 5. Найти вторую производную от функции:

$$y = \frac{x+1}{\sqrt{1 - x - x^2}}.$$

Задание 6. Вычислить определенный интеграл:

$$\int_{\pi/7}^{\pi/4} \frac{\cos(2x) + \sin^2(x)}{\sin(3x)} dx.$$

Задание 7. Вычислить двойной интеграл:

$$\int_0^1 \int_1^2 x^2 \cdot \exp(x + \sin(y)) \cdot \cos(y) dx dy.$$

Задание 8. Вычислить неопределенный интеграл:

$$\int \frac{\ln x}{x} dx.$$

Задание 9. Решить дифференциальное уравнение и построить график функции y(x) на отрезке [1,10]:

$$x^2 \cdot y'' + 3xy' + y = 1/x$$
$$y(1) = 1$$
$$y'(1) = 0$$

Задание 10. Решить дифференциальное уравнение:

$$y' = \exp(x + y) + \exp(x - y).$$

Контрольные вопросы

1. Что называют операцией правого и левого деления матриц?
2. Как задать функцию пользователя в системе MATLAB?
3. Как можно приближенно определить нули функции?
4. Как можно достигнуть большей точности при нахождении нулей функции?
5. Как определяются корни многочлена в системе MATLAB?
6. Как вычислить определенный интеграл и двойной интеграл в системе MATLAB?
7. Опишите схему нахождения решений системы дифференциальных уравнений с начальными условиями?
8. Как произвести упрощение алгебраического выражения в системе MATLAB?
9. Как символьно определить производную n-ого порядка от явно и неявно заданных функций?
10. Опишите функцию *dsolve*.

Приложение 1

Стандартные функции вещественного аргумента

Экспоненциальные функции

a^x	Степенная функция
x^a	Показательная функция
sqrt(x)	Квадратный корень
exp(x)	Экспонента
log(x)	Натуральный логарифм
log10(x)	Десятичный логарифм
abs(x)	Модуль
fix(x)	Отбрасывание дробной части числа
floor(x)	Округлениие до меньшего целого
ceil(x)	Округлениие до большего целого
round(x)	Обычное округление
rem(x,y)	Остаток от делениях x на y без учёта знака
mod(x,y)	Остаток от делениях x на y с учёта знака
sign(x)	Знак числа
factor(x)	Разложение числа x на простые множители

Тригонометрические функции

sin(x)	Синус
sinh(x)	Синус гиперболический
asin(x)	Арксинус
asinh(x)	Арксинус гиперболический
cos(x)	Косинус
cosh(x)	Косинус гиперболический
acos(x)	Арккосинус
acosh(x)	Арккосинус гиперболический
tan(x)	Тангенс
tanh(x)	Тангенс гиперболический
atan(x)	Арктангенс

atanh(x)0	Арктангенс гиперболический
cot(x)	Котангенс
coth(x)	Котангенс гиперболический
acot(x)	Арккотангенс
acoth(x)	Арккотангенс гиперболический

Приложение 2

Системные переменные MATLAB

i, j	Мнимая единица
pi	Число π
eps	Погрешность для операций над числами с плавающей точкой (по умолчанию 2^{-52})
realmin	Минимальное значение вещественного числа
realmax	Максимальное значение вещественного числа
inf	Бесконечность
NaN	Неопределённость
ans	Переменная, хранящая результат последней операции

Приложение 3

Функции комплексных переменных

abs	Абсолютное значение комплексного числа
conj	Комплексно-сопряжённое число
imag	Мнимая часть комплексного числа
real	Действительная часть комплексного числа
angle	Аргумент комплексного числа
isreal	“Истина”, если число действительное

Литература

1. Мартынов Н.Н., Иванов А.П. MATLAB 5.X. Вычисления, визуализация, программирование. – М.:КУДИЦ-ОБРАЗ, 2000. – 336 с.
2. Дьяконов В.П., Абраменкова И.В. MATLAB 5.0/5.3. Система символьной математики. – М.: Нолидж, 1999. – 640 с., ил.
3. Говорухин В., Цибулин В. Компьютер в математическом исследовании. Учебный курс. – СПб.: Питер, 2001. – 624 с., ил.
4. Глушаков С.В., Жакин И.А., Хачиров Т.С. Математическое моделирование: Mathcad 2000, MATLAB 5. Учебный курс. – Харьков.: Фолио, 2001. –524 с.
5. Мартынов Н**.** Введение в MatLab 6. – М.: Кудиц-образ, 2002.
6. Дьяконов В**.** Matlab6. Учебный курс. – СПб.: Питер, 2001.
7. Потемкин В**.** Введение в MATLAB. – М.: Диалог-МИФИ, 2000