ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Высшая математика»

ВЫСШАЯ МАТЕМАТИКА. ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА. МАТЕМАТИКА. МАТЕМАТИКА (СПЕЦГЛАВЫ). МАТЕМАТИЧЕСКИЕ ОСНОВЫ ТЕОРИИ ПРИНЯТИЯ РЕШЕНИЙ. ОСНОВЫ КОМБИНАТОРИКИ. ПАКЕТ ПРИКЛАДНЫХ ПРОГРАММ MATLAB ДЛЯ ИССЛЕДОВАНИЙ И РАЗРАБОТОК. ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ ДЛЯ АНАЛИЗА ДАННЫХ. ПРИКЛАДНАЯ МАТЕМАТИКА. СПЕЦГЛАВЫ МАТЕМАТИКИ. ТЕОРИЯ ВЕРОЯТНОСТЕЙ И МАТЕМАТИЧЕСКАЯ СТАТИСТИКА. ТЕОРИЯ ВЕРОЯТНОСТЕЙ, МАТЕМАТИЧЕСКАЯ СТАТИСТИКА И СЛУЧАЙНЫЕ ПРОЦЕССЫ

Методические рекомендации к практическим занятиям для студентов всех специальностей, обучающихся по белорусским и российским образовательным программам, дневной и заочной форм обучения

ПАКЕТ ПРИКЛАДНЫХ ПРОГРАММ MATLAB

Могилёв 2017

УДК 004.4
ББК 32.973
В 93

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Высшая математика» «25» мая 2017 г., протокол № 9

Составители: ст. преподаватель А. Г. Козлов;
канд. физ.-мат. наук, доц. Д. В. Роголев

Рецензент И. Д. Камчицкая

Приведены основные теоретические сведения и описаны приёмы работы с пакетом прикладных программ Matlab.

Учебно-методическое издание

ВЫСШАЯ МАТЕМАТИКА. ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА. МАТЕМАТИКА. МАТЕМАТИКА (СПЕЦГЛАВЫ). МАТЕМАТИЧЕСКИЕ ОСНОВЫ ТЕОРИИ ПРИНЯТИЯ РЕШЕНИЙ. ОСНОВЫ КОМБИНАТОРИКИ. ПАКЕТ ПРИКЛАДНЫХ ПРОГРАММ MATLAB ДЛЯ ИССЛЕДОВАНИЙ И РАЗРАБОТОК. ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ ДЛЯ АНАЛИЗА ДАННЫХ. ПРИКЛАДНАЯ МАТЕМАТИКА. СПЕЦГЛАВЫ МАТЕМАТИКИ. ТЕОРИЯ ВЕРОЯТНОСТЕЙ И МАТЕМАТИЧЕСКАЯ СТАТИСТИКА. ТЕОРИЯ ВЕРОЯТНОСТЕЙ, МАТЕМАТИЧЕСКАЯ СТАТИСТИКА И СЛУЧАЙНЫЕ ПРОЦЕССЫ

Ответственный за выпуск В. Г. Замураев

Технический редактор А. А. Подошевко

Компьютерная вёрстка Е. С. Лустенкова

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 56 экз. Заказ №

Издатель и полиграфическое исполнение:
Государственное учреждение высшего профессионального образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилев.

Содержание

1 Ознакомление с системой Matlab

1.1 Общие сведения о системе Matlab

Matlab (**Mat**rix **Lab**oratory – матричная лаборатория) – универсальная интегрированная система, предлагаемая её разработчиками как **язык программирования высокого уровня для технических вычислений.**

Язык программирования Matlab является интерпретатором. Это значит, что каждая инструкция программы распознается и тут же исполняется. Этап компиляции полной программы отсутствует. Интерпретация означает, что Matlab не создаёт исполняемых конечных программ. Программы существуют лишь в виде m-файлов (файлов с расширением m). Для выполнения программ необходимо находиться в среде Matlab. Однако разработчиками Matlab созданы компиляторы, транслирующие программы на языке Matlab в коды языков программирования C и C++. Это решает проблему создания исполняемых программ, изначально написанных в среде Matlab.

1.2 Интерфейс системы Matlab

Matlab запускается любым стандартным для Windows способом. После этого появляется окно системы Matlab, и система готова к проведению вычислений в командном режиме. Полезно знать, что в начале запуска автоматически выполняется команда **matlabrc**, которая исполняет загрузочный файл **matlabrc.m** и файл **startup.m**, если таковой существует. Эти файлы выполняют начальную настройку терминала системы и задают ряд её параметров. Для сохранения собственных m-файлов рекомендуется создать пользовательский каталог, например, каталог с именем USER на диске D. Доступ к этому каталогу необходимо обеспечить с помощью команды **path**, которая будет иметь вид:

path(path,'D:\USER')

Данную команду целесообразно включить в файл **startup.m,** который, в свою очередь, нужно создать и записать в один из каталогов системы Matlab, например, в каталог, в котором размещается файл **matlabrc.m.**

Окно Matlab состоит из трёх основных частей (рисунок 1.1):

1) Current folder – текущий каталог проекта;
2) Command Window – окно для ввода команд и отображения результатов;
3) Workspace – окно, в котором будут отображаться созданные переменные и функции.

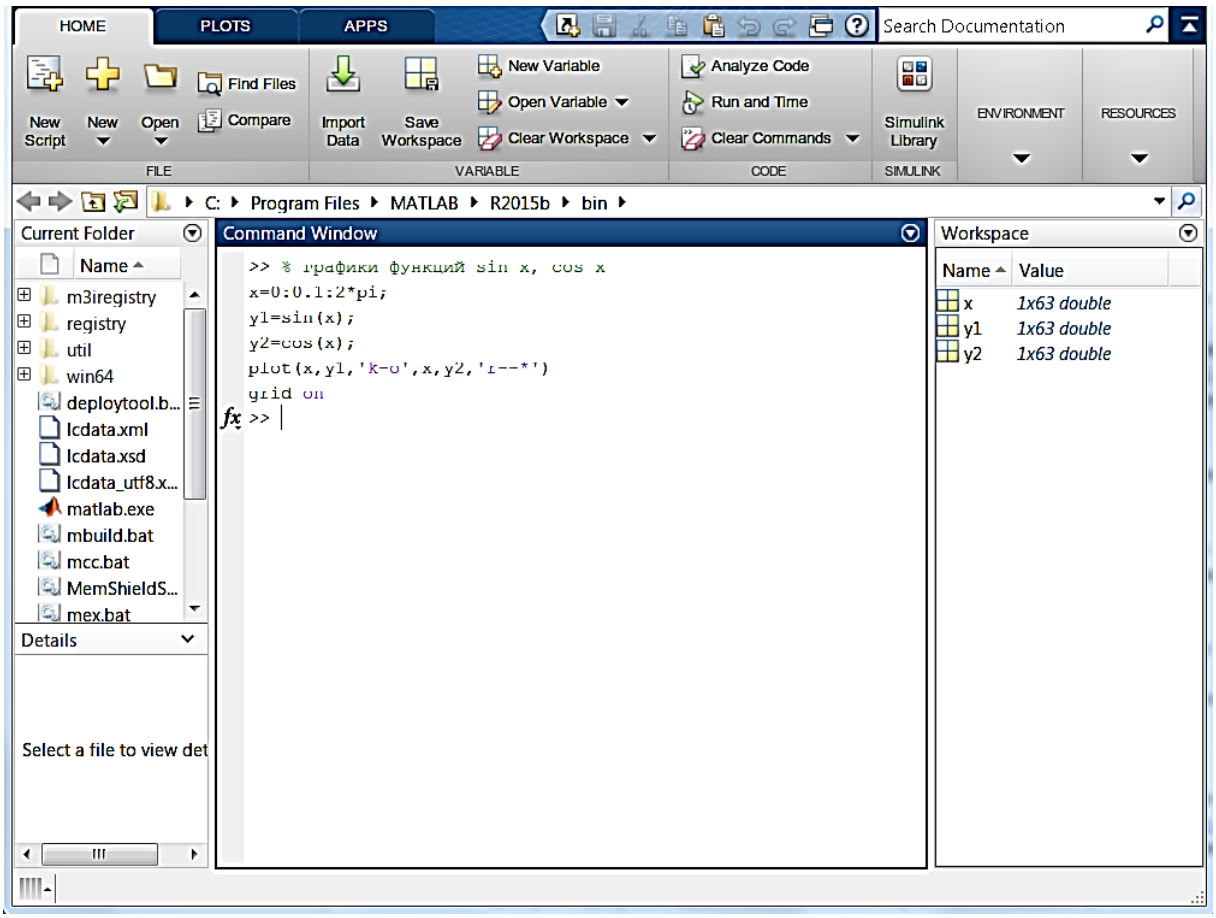

Рисунок 1.1 – Основное окно Matlab

Строка ввода указывается с помощью приглашающего символа >>. В строке вывода символ >> отсутствует. Строка сообщений об ошибках начинается символами ???. Команды набираются на клавиатуре с помощью обычных операций строчного редактирования. Особое назначение имеют клавиши ↑ и ↓. Они используются для подстановки после приглашения >> ранее введённых строк, например, для их дублирования, исправления или дополнения.

Полезно запомнить следующие команды:

clc – очищает экран и размещает курсор в левом верхнем углу пустого экрана;

clear – уничтожает в рабочем пространстве определения всех переменных;

clear x – уничтожает в рабочем пространстве определение переменной x;

clear a,b,c – уничтожает в рабочем пространстве определения переменных списка.

Уничтоженная (стёртая в рабочем пространстве) переменная становится неопределённой. Использовать такие переменные нельзя, попытки сопровождаются выдачей сообщений об ошибке. По мере задания одних переменных и уничтожения других рабочая область перестаёт быть непрерывной и содержит «дыры» и всякий «мусор». Во избежание непроизводительных потерь памяти при работе с объёмными данными следует использовать команду **pack**, осуществляющую дефрагментацию рабочей области.

Все данные, вводимые и полученные пользователем во время сессии, хранятся в особой области памяти, называемой **рабочей областью**. При завершении сессии рабочая область очищается, т.е. все данные пропадают, к ним нельзя обратиться во время новой сессии.

Для сохранения рабочей области служит команда **save <имя_файла>** или кнопка **Save Workspace**, после выполнения которой вся рабочая область записывается на диск в бинарном формате с именем <имя_файла>.mat. Для загрузки рабочей области ранее проведенной (и сохраненной) сессии служит команда **load <имя_файла>** или кнопка **Open.**

1.3 Ввод команд в Matlab

Система Matlab ориентирована на работу с матричными переменными. По умолчанию предполагается, что каждая заданная переменная – это матрица. Даже обычные константы и переменные рассматриваются в Matlab как матрицы размером 1×1. Например, если ввести X=1, то система воспринимает это как задание матрицы с единственным элементом, значение которого равно 1. В свою очередь, вектор – это матрица, число столбцов которой равно 1.

Такой подход позволяет в общем случае одной командой обращаться сразу к множеству элементов матрицы (массива) без использования циклов.

Простейшей конструкцией языка Matlab является оператор присваивания:

Имя_переменной = Выражение

Типы переменных заранее не декларируются. Они определяются выражением, значение которого присваивается переменной. Так, если это выражение – вектор или матрица, то переменная будет векторной или матричной.

После набора оператора в командной строке и нажатия клавиши ENTER на экран выводится вычисленное значение переменной. Для блокировки вывода результата вычислений на экран оператор нужно завершить символом ; (точка с запятой).

Пример

```
>> x=2;
>> y=2;
>> r=sqrt(x^2+y^2)
r=
2.8284
```

Возможна также конструкция, состоящая только из выражения. В этом случае для результата вычислений Matlab автоматически назначает переменную с именем **ans.**

Пример

```
>> x=2;
>> y=2;
>> sqrt(x^2+y^2)
ans=
2.8284
```

Действительные и комплексные числа.

Число – простейший объект языка MATLAB, представляющий количественные данные. Числа можно считать константами. Числа используются в общепринятом представлении о них. Они могут быть целыми, дробными, с фиксированной и плавающей точкой. Возможно представление чисел в хорошо известном научном формате с указанием мантиссы и порядка числа. Далее приводятся примеры представления действительных чисел:

```
0
-3
2.301
123.456e-24
-234.456e10
```

Как нетрудно заметить, в мантиссе чисел целая часть отделяется от дробной не запятой, а точкой, как принято в большинстве языков программирования. Для отделения порядка числа от мантиссы используется символ e. Знак «плюс» у чисел не проставляется, а знак «минус» у числа называют *унарным минусом*. Пробелы между символами в числах не допускаются.

Числа могут быть *комплексными*: z=Re(x)+Im(x)*i. Такие числа содержат действительную Re(z) и мнимую Im(z) части. Мнимая часть имеет множитель i или j, означающий корень квадратный из –1:

```
3i
2j
2+3i
-3.141i
-123.456+2.7e-3i
```

Функция real(z) возвращает действительную часть комплексного числа, Re(z), а функция imag(z) – мнимую, Im(z). Для получения модуля комплексного числа используется функция abs(z), а для вычисления аргумента – angle(Z).

Форматы чисел.

Для установки определённого формата представления чисел используется команда

`format name`,

где name – имя формата. Для иллюстрации различных форматов рассмотрим вектор, содержащий два элемента-числа:

x=[4/3 1.2345e-6]

В различных форматах их представления будут иметь следующий вид:

```
format short 1.3333 0.0000
format short e 1.3333E+000 1.2345E-006
format long 1.333333333333338 0.000001234500000
```

```
format long e 1.333333333333338E+000 1.234500000000000E-006
format bank 1.33 0.00
```

Отметим, что задание формата сказывается только на форме вывода чисел. Вычисления все равно происходят в формате двойной точности, а ввод чисел возможен в любом удобном для пользователя виде.

Константы и системные переменные.

Константа – это предварительно определённое числовое или символьное значение, представленное уникальным именем (идентификатором). Числа (например, 1, –2 и 1.23) являются безымянными числовыми константами.

Другие виды констант в MATLAB принято называть системными переменными, поскольку, с одной стороны, они задаются системой при её загрузке, а с другой – могут переопределяться. Основные системные переменные, применяемые в системе MATLAB, указаны далее:

i или j –мнимая единица (корень квадратный из –1);

pi – число π = 3,1415926…;

eps – погрешность операций над числами с плавающей точкой (2^{-52});

realmin – наименьшее число с плавающей точкой (2^{-1022});

realmax – наибольшее число с плавающей точкой (2^{1023});

inf – значение машинной бесконечности;

ans – переменная, хранящая результат последней операции и обычно вызывающая его отображение на экране дисплея;

NaN – указание на нечисловой характер данных (Not-a-Number).

1.4 Операторы и встроенные функции Matlab

Оператор – это специальное обозначение для определённой операции над данными – операндами. Например, простейшими арифметическими операторами являются знаки суммы +, вычитания –, умножения * и деления /. Операторы используются совместно с операндами. Например, в выражении 2+3 знак + выступает оператором сложения, а числа 2 и 3 – операндами. Операторы также являются распространёнными объектами математических выражений и языков программирования.

Для выполнения арифметических операций в системе Matlab применяются обычные символы: + (сложение), – (вычитание), * (умножение), / (деление), ^ (возведение в степень). Эти операции называются матричными, так как применяются и при работе с матрицами. Наряду с матричными операциями над массивами можно выполнять и поэлементные операции. Для обозначения поэлементных операций используется символ . (точка), предшествующий обычной (матричной) операции.

Следует отметить, что большинство операторов относятся к матричным операциям, что может служить причиной серьёзных недоразумений. Например, операторы умножения * и деления / вычисляют произведение и частное от деления двух массивов, векторов или матриц. Есть ряд специальных операторов,

например оператор \ означает деление справа налево, а операторы .* и ./ поэлементное умножение и поэлементное деление массивов соответственно.

Следует отметить, что в математических выражениях системы Matlab операторы имеют определённый **приоритет исполнения**:

1) круглые скобки;
2) операции транспонирования и возведения в степень;
3) одноместные операции (унарные + и –, логическое отрицание ~);
4) арифметические операции умножения и деления;
5) арифметические операции сложения и вычитания;
6) оператор сечения массива :;
7) операторы отношения;
8) логические операторы и т. д.

Для изменения приоритета операций в математических выражениях используются круглые скобки. Степень вложения скобок не ограничивается.

Следующие примеры поясняют сказанное на примере операций с векторами:

```
>> V1=[2 4  6  8]
V1 =    2   4  6   8
>> V2=[1 2  3  4]
V2 =    1   2  3   4
>> V1/V2
ans =   2.0000
>> V1.*V2
ans =   2   8  18 32
>> V1./V2
ans =   2   2  2   2
```

Полный список операторов можно получить, используя команду **help ops**.

Функции – это имеющие уникальные имена объекты, выполняющие определенные преобразования своих аргументов и при этом возвращающие результаты этих преобразований. Возврат результата – отличительная черта функций. При этом результат вычисления функции с одним выходным параметром подставляется на место её вызова, что позволяет использовать функции в математических выражениях, например, функцию sin в 2*sin(pi/2).

Функции в общем случае имеют список аргументов (параметров), заключённый в круглые скобки. Например, функция Бесселя записывается как bessel(NU,X). В данном случае список параметров содержит два аргумента – NU в виде скаляра и X в виде вектора. Многие функции допускают ряд форм записи, отличающихся списком параметров. Если функция возвращает несколько значений, то она записывается в виде

[Y1, Y2,...]=func(X1, X2,...),

где Y1, Y2,... – список выходных параметров и X1, X2,... – список входных аргументов (параметров).

Со списком элементарных функций можно ознакомиться, выполнив команду **help elfun**, а со списком специальных функций – с помощью команды **help specfun**. Функции могут быть встроенными (внутренними) и внешними, или m-функциями. Так, встроенными являются наиболее распространённые элементарные функции, например, sin(x) и exp(y), тогда как функция sinh(x) – внешняя. Внешние функции содержат свои определения в m-файлах. Задание таких функций возможно с помощью специального редактора m-файлов. Встроенные функции хранятся в откомпилированном ядре системы Matlab, в силу чего они выполняются предельно быстро.

Функции пользователя.

Хотя ядро новых версий системы Matlab содержит уже более 1000 встроенных функций (не считая функций, определённых в десятках пакетов расширения), всегда может понадобиться какая-то нужная пользователю функция. Язык программирования системы Matlab предоставляет ряд возможностей для задания функций пользователя. Одна из таких возможностей заключается в применении функции **inline**, аргументом которой надо в апострофах задать выражение, задающее функцию одной или нескольких переменных. В приведённом далее примере задана функция двух переменных – суммы квадратов sin(x) и cos(y):

```
>> sc2=inline('sin(x).^2+cos(y).^2')
sc2 =
Inline function:
sc2(x,y) = sin(x).^2+cos(y).^2
```

Для присваивания значений массиву необходимо значения элементов массива перечислить в квадратных скобках, разделяя их пробелами или запятыми.

Пример

```
>> v=[1 5 3]
v=
1 5 3
```

В этом примере задан вектор v (одномерный массив) со значениями элементов 1, 5, 3. Задание матрицы (двухмерного массива) требует указания различных строк. Для различения строк используется ; (точка с запятой).

Пример

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

Для обращения к отдельному элементу массива используются имя массива и круглые скобки, внутри которых указываются индексы, разделённые запятыми. Отметим, что индексация (нумерация) элементов, строк и столбцов начинается с единицы.

Пример

```
>> m=[1 2 3; 4 5 6; 7 8 9];
>> m(1,1)=5;
>> m(3,3)=m(1,1)+m(3,3);
>> m
m=
5  2   3
4  5   6
7  8   14
```

Matlab допускает максимум 4096 символов в строке. Если для выражения не хватает одной строки или нет желания заходить в невидимую область окна, то выражение можно перенести на новую строку с помощью многоточия … (3 или более точек). Комментарий в строке должен начинаться символом %.

Пример

```
>> % Пояснение переноса выражения и комментариев
>> x=2;
>> y=2;
>> r=sqrt(x^2+ … % перенос выражения в следующую строку
y^2)
r=
    2.8284
```

Для формирования упорядоченных числовых последовательностей в Matlab применяется оператор : (двоеточие):

Начальное_значение: Шаг: Конечное_значение

Данная конструкция порождает последовательность (массив) чисел, которая начинается с начального значения, идёт с заданным шагом и завершается конечным значением. Если шаг не задан, то он принимает значение 1 или –1.

Пример

```
>> i=1:6
i=
1 2 3 4 5 6
>>x=0:0.5:3
x=
0  0.5000  1.0000  1.5000  2.0000  2.500  3.0000
>> x=3:-0.5:0
x=
3.000  2.5000  2.0000  1.5000  1.0000  0.5000   0
```

Сортировка.

Команда **y=sort(x)** сортирует элементы вектора $\mathbf{x}=(x_1,x_2,\ldots,x_n)$ в возрастающем порядке. Здесь **x** – исходный вектор, **y** – отсортированный вектор. В случае, когда **x –** матрица, функция **y=sort(x)** сортирует каждый столбец **x** в возрастающем порядке.

y=sort(x,d) сортирует матрицу **x** вдоль измерения **d**.

[y,i]=sort(x,d) возвращает также индексную матрицу **i**. Если **x** – вектор, то элементы индексной матрицы указывают номера элементов вектора **y** в исходном векторе **x.**

Программа сортировки используется для формирования вариационного ряда из имеющейся выборки.

Задание

1 Определите последовательность значений экспоненты e^{at} при выбранном вами значении действительного параметра $a\,(a<0)$ и таком диапазоне изменения времени t, который позволит получить качественное представление о виде зависимости e^{at} при небольшом числе (не больше десяти) дискретных значений переменной t.

2 Проделайте то же самое при чисто мнимом значении параметра $a=i\omega$.

3 Проделайте то же самое для функции, представленной суммой двух экспонент с комплексно сопряжёнными показателями $a_1=i\omega$, $a_2=-i\omega$.

4 Проделайте то же самое для функции, представленной суммой двух экспонент с комплексно сопряжёнными показателями $a_1=-b+i\omega$, $a_2=-b-i\omega$.

5 Проделайте то же самое для функций $y=\cos(\omega t)$, $y=e^{-bt}\cos(\omega t)$.

1.5 Справочная система Matlab

Matlab имеет подробную справочную систему, которая активизируется щелчком левой клавиши мыши на пункте **?** главного меню Matlab. Справочная система позволяет ознакомиться с языком программирования Matlab, имеющимися в системе функциями, их назначением и описанием.

Также работа со справкой может осуществляться из командной строки.

В окне справки отображается первая страница встроенного справочника системы Matlab, на которой приведён список разделов библиотеки функций Matlab. Чтобы получить полную информацию по интересующему разделу, необходимо установить курсор на соответствующую строку и дважды нажать левую кнопку мыши. В поле просмотра появится список функций, входящих в данный раздел, с указанием назначения каждой из них. Выбрав определённую функцию и дважды нажав левую кнопку мыши, можно получить более подробную информацию об этой функции.

При первом ознакомлении с системой – это самый естественный способ получения справки о функциях. Основным же и более быстрым способом получения информации о синтаксисе и способе использования конкретной функции является применение команды **help** <*имя функции*>.

Если нет уверенности в правильном написании имени функции или оно, вообще, не известно, можно воспользоваться поиском по ключевому слову (или последовательности таких слов). Для этого предназначена команда

lookfor Ключевое слово

или

lookfor 'Ключевые слова'

Команда **doc** или пункт Documentation меню Help позволяет получить доступ к большому объёму справочной информации и к документации по системе. Все операторы и функции системы Matlab описаны в формате HTML более подробно, чем во встроенном справочнике (к которому можно получить доступ по команде **help**). Использование более детальной справочной информации, чем та, которую можно получить по команде **help**, необходимо достаточно подготовленному пользователю. Начинающему пользователю в первую очередь нужны руководства по функциям системы, языку программирования, графическим функциям и т. п.

Команда **demo** или пункт **Examples** меню **Help** предоставляет доступ к галерее примеров системы Matlab. После выполнения этой команды появляется окно примеров **Matlab Examples.** Здесь содержится большое число достаточно серьёзных примеров применения системы. При необходимости можно ознакомиться с файлами примеров и даже перенести тексты программ в командное окно Matlab, используя буфер промежуточного хранения.

Задание

1 Выясните, каким пунктам справочного меню соответствуют результаты ввода следующих команд:

```
doc
helpwin
demo
```

2 Ознакомьтесь с универсальными командами, операторами, конструкциями языка *программирования*, элементарными матрицами и математическими функциями.

3 В меню Help выберите пункт «Examples» и ознакомьтесь с галереей примеров.

2 Построение графиков

2.1 Построение графиков функций одной переменной

Для построения графиков функций одной переменной $y = f(x)$ в Matlab имеется функция **plot.** График строится в декартовой системе координат по заданным массивам значений аргумента и функции. Заданные этими массивами точки соединяются прямыми линиями. Имеется возможность изменять тип и цвет линии и тип узловых точек (маркер). Вызов этой функции осуществляется командой **plot(x,y,s).**

Здесь **x, y** – одномерные массивы одинаковой размерности; **x** – массив значений аргумента функции $y = f(x)$; **y** – массив значений функции $y = f(x)$; **s** – строковая константа, определяющая цвет линии, маркер узловых точек и тип линии. Эта константа может содержать от одного до трёх символов.

Цвет линии определяется символами **y** (жёлтый), **m** (фиолетовый), **c** (голубой), **r** (красный), **g** (зелёный), **b** (синий), **w** (белый), **k** (чёрный).

Тип узловой точки определяется символами **.** (точка), **o** (окружность),

x (крестик), **+** (плюс), ***** (звёздочка), **s** (квадрат), **d** (ромб), **< > ^** (треугольники различной направленности), **p** (пятиугольник), **h** (шестиугольник).

Тип линии определяется символами - (непрерывная), : (короткие штрихи), -. (штрихпунктир), -- (длинные штрихи).

Символьную константу **s** можно опустить. В этом случае по умолчанию используется непрерывная линия жёлтого цвета.

Для построения в одном окне нескольких графиков можно использовать команду

plot(x1,y1,s1,x2,y2,s2,x3,y3,s3,…)

Пример

```
% графики функций sin x, cos x
x=0:0.1:2*pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,'k-o',x,y2,'r--*')
grid on
```

В результате выполнения этой программы на экран монитора будет выведено графическое окно с графиками, отображёнными на рисунке 2.1. Графики представлены в черно-белой палитре, хотя в действительности график функции cos(x) выводится красным цветом.

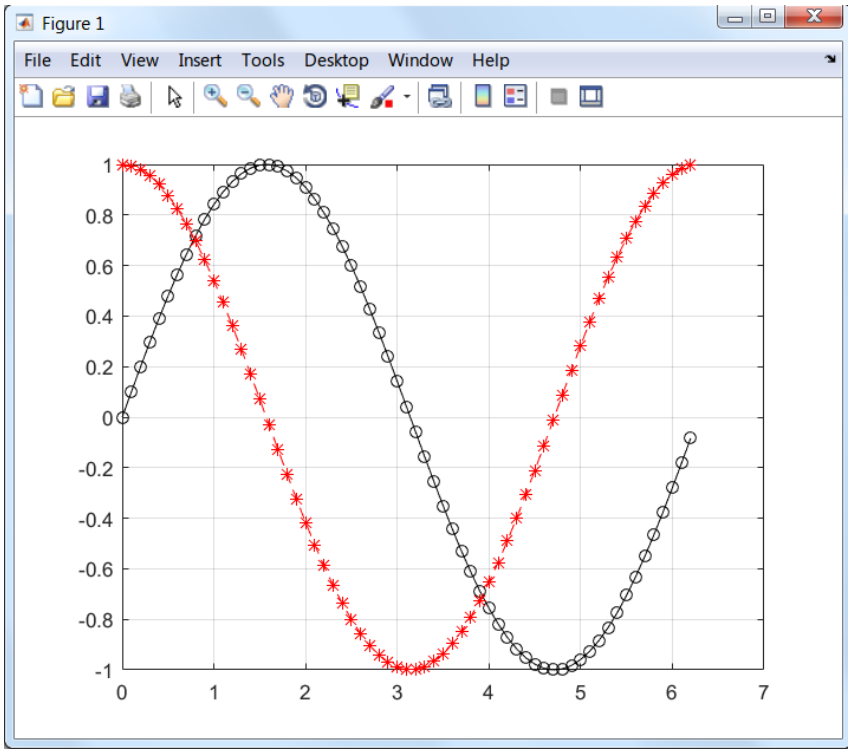

Рисунок 2.1 – Графики, выполненные с помощью программы plot

Команда **grid on** добавляет на график сетку.

Созданный график можно скопировать в буфер Clipboard, активизировав в

пункте **Edit** главного меню окна графики команду **Copy Figure**, с целью его дальнейшего редактирования в каком-либо графическом редакторе, например Paint.

2.2 Создание массивов трёхмерной графики

Трёхмерные поверхности обычно описываются функцией двух переменных $z = f(x, y)$. Специфика построения трёхмерных графиков в Matlab требует не просто задания ряда значений x и y, то есть векторов **x** и **y**, а определения двухмерных массивов **X** и **Y**. Для создания таких массивов служит функция **meshgrid.**

[X,Y]=meshgrid(x,y) – преобразует область, заданную векторами **x** и **y**, в двухмерные массивы **X** и **Y**, которые могут быть использованы для вычисления значений функции двух переменных и построения трёхмерных графиков. Эта функция формирует массивы **X** и **Y** таким образом, что строки выходного массива **X** являются копиями вектора **x**, а столбцы выходного массива **Y** – копиями вектора **y**.

Пример

```
[X,Y]=meshgrid(1:1:4,6:1:9)
X =
     1     2     3     4
     1     2     3     4
     1     2     3     4
     1     2     3     4
Y =
     6     6     6     6
     7     7     7     7
     8     8     8     8
     9     9     9     9
```

В этом примере формируются массивы X и Y для построения трёхмерной поверхности при изменении *x* от 1 до 4 с шагом 1 и *y* от 6 до 9 с шагом 1.

2.3 Построение контурных графиков

Контурные графики представляют собой проекции на плоскость xOy сечений функции двух переменных $z = f(x, y)$ горизонтальными плоскостями. Контурные графики называют также линиями равного уровня, которые получаются, если трёхмерная поверхность пересекается рядом горизонтальных плоскостей, расположенных параллельно друг другу.

Для построения контурных графиков используется команда **contour**:

contour(x,y,z,n) строит контурный график по данным матрицы **z** с указанием спецификаций для **x** и **y** с заданием **n** линий равного уровня;

contour(x,y,z,v) строит линии равного уровня для высот, указанных значениями элементов вектора **v**.

Пример

```
[x,y]=meshgrid(-3: .2: 3, -3: .2: 3);
z=x.^2+y.^2;
contour(x,y,z,8)
%или contour(x,y,z,[4,4])
grid on
```

По этой программе на экран монитора будет выведено восемь графиков, представленных на рисунке 2.2.

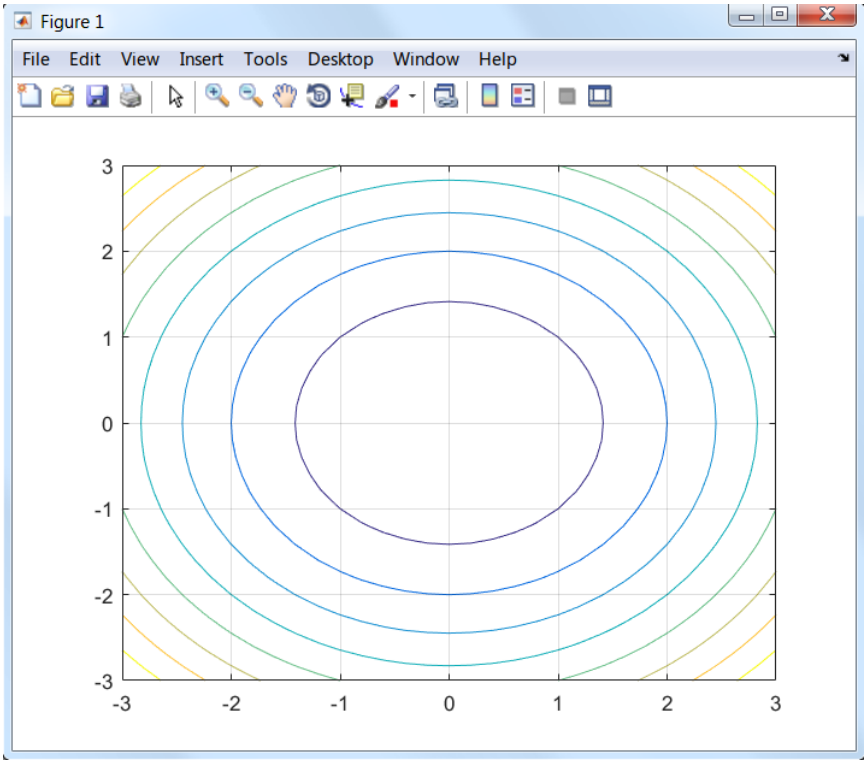

Рисунок 2.2 – Контурные графики, построенные с помощью функции **contour**

2.4 Построение графиков трёхмерных поверхностей

Команда **plot3(…)** является аналогом команды **plot(…)**, но относится к функции двух переменных $z = f(x, y)$. Она строит изображение трёхмерных (3D) поверхностей.

plot3(X,Y,Z) в случае, когда **X,Y,Z** – векторы одинаковой длины n, вычерчивает линию в трёхмерном пространстве через точки, координатами которых являются элементы векторов **X,Y,Z,** т. е. через точки $z_i = f(x_i, y_i)$, $i = \overline{1,n}$. В случае, когда **X,Y,Z –** двухмерные массивы **(**матрицы) одинаковых размеров, эта функция вычерчивает несколько линий, полученных из столбцов **X,Y,Z.** Массивы **X** и **Y** можно получить с помощью функции **meshgrid**.

Пример

```
[x,y]=meshgrid(-2:0.1:2,-2:0.1:2);
z=x.^2+y.^2;
```

```
plot3(x,y,z)
grid on
```

По этой программе будет выведена фигура, представленная на рисунке 2.3.

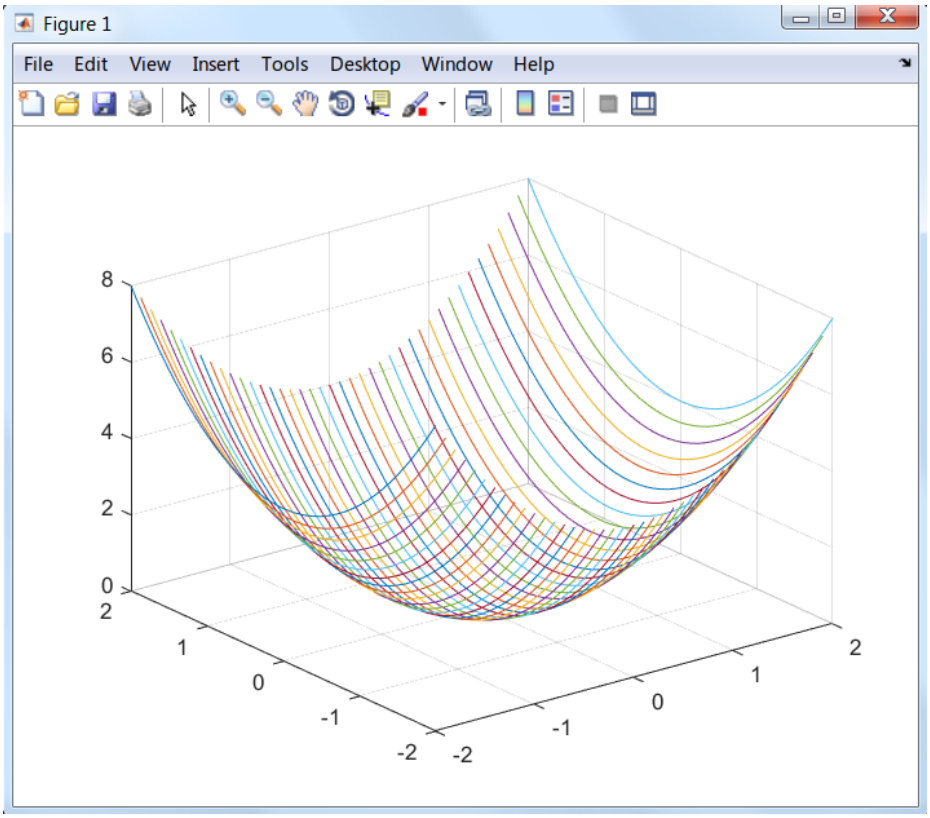

Рисунок 2.3 – Трёхмерный график, построенный с помощью функции **plot3**

Однако более наглядными являются сеточные графики трёхмерных поверхностей с заданной или функциональной окраской. Такие графики выполняются командой **mesh**:

mesh(X,Y,Z,C) – выводит в графическое окно сетчатую поверхность с цветами узлов поверхности, заданных массивом **C**;

mesh(X,Y,Z) – аналог предшествующей команды при **C=Z** с использованием функциональной окраски, при которой цвет задаётся высотой поверхности.

Пример

```
[x,y]=meshgrid(-2:0.1:2,-2:0.1:2);
z=x.^2+y.^2;
mesh(x,y,z)
```

В результате выполнения этой программы на экран будет выведена фигура, представленная на рисунке 2.4.

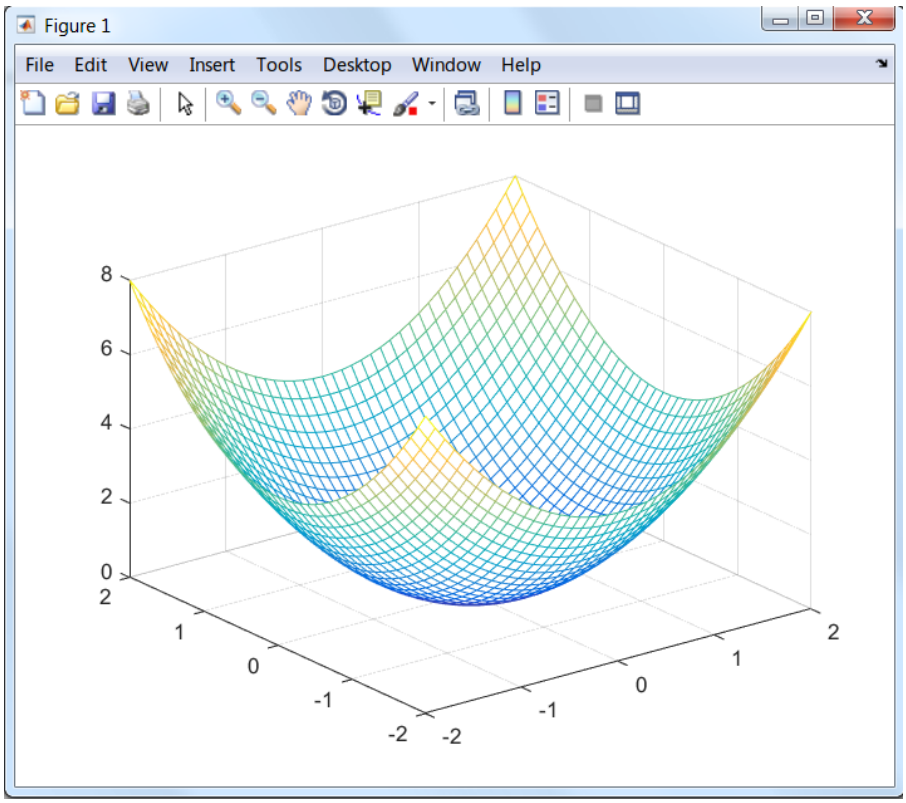

Рисунок 2.4 – Трёхмерный график, построенный с помощью функции **mesh**

Если возникают трудности с использованием поэлементных операций при формировании массива **z** значений функции, то этот массив можно сформировать без применения поэлементных операций с помощью двух вложенных циклов **for**. Так, предыдущий пример можно оформить следующим образом:

Пример

```
x1=[-2:0.1:2];
x2=[-2:0.2:2];
nx1=length(x1);
nx2=length(x2);
[x,y]=meshgrid(x1,x2);
for i=1:nx1
  for j=1:nx2
  z(j,i)=x1(i)^2+x2(j)^2;
end;
end;
mesh(x,y,z)
```

Во многих случаях желательно построение ряда наложенных друг на друга графиков в одном и том же окне. Такую возможность обеспечивает команда продолжения графических построений **hold**:

hold on обеспечивает продолжение вывода графиков в текущее окно графики, что позволяет добавлять последующие графики к уже существующему;

hold off отменяет режим продолжения графических построений.

Пример

```
t=-3:0.2:3;
[x,y]=meshgrid(t,t);
z=x.^2+y.^2;
contour(x,y,z,8)
hold on
v=2*t;
plot(t,v,'k.-')
grid on
hold off
```

В этом примере на контурный график функции $z = x^2 + y^2$ наносится график прямой линии $y = 2x$ (рисунок 2.5).

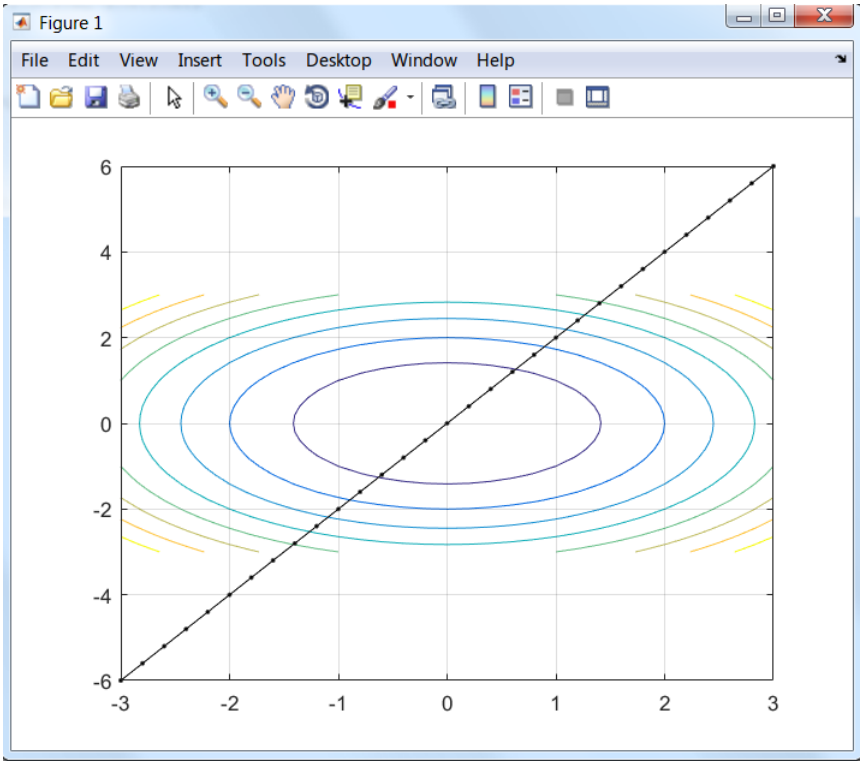

Рисунок 2.5 – Два графика в одном графическом окне

3 Основы программирования в Matlab

3.1 Создание m-файлов

В Matlab имеется возможность написать программу и сохранить её в виде **m-файла-сценария** с целью последующего многократного выполнения. Этот файл, именуемый также **script-файлом**, представляет собой, в отличие от **m-файлов-функций**, последовательность команд без входных и выходных параметров. Он имеет следующую структуру:

% Основной комментарий
%Дополнительный комментарий
Тело файла с любыми выражениями

Созданный m-файл включается в справочную систему. Комментарии в m-файле нужны для того, чтобы ознакомиться с назначением файла через справочную систему. Основной комментарий выводится при исполнении команд **lookfor** и **help имя_каталога**. Полный комментарий выводится при исполнении команды **help имя_файла**.

Для создания и отладки m-файлов необходимо войти в редактор-отладчик Matlab, выбрав на панели Matlab пункт **New script** (горячие клавиши CTRL+N) либо **New function**. Откроется новая вкладка редактора-отладчика (рисунок 3.1).

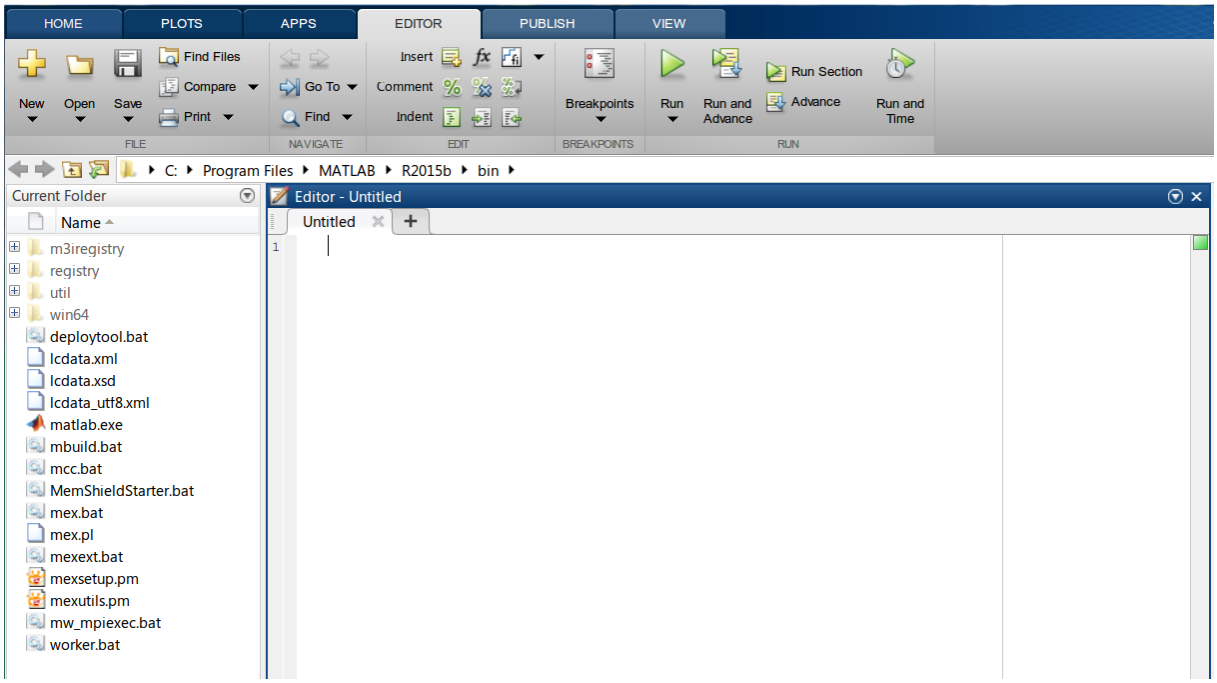

Рисунок 3.1 – Редактор m-файлов

m-файл-функция является типичным объектом языка программирования системы Matlab. Структура m-файла-функции *с одним выходным параметром* выглядит следующим образом:

function var=f_name(список параметров)
% Основной комментарий
%Дополнительный комментарий
Тело файла с любыми выражениями
var=выражение

Здесь переменная **var** – выходной параметр, **f_name** – имя функции.

Функция возвращает свое значение **var** и может использоваться в математических выражениях в виде **f_name(список параметров)**.

Все переменные, имеющиеся в теле файла-функции, являются *локальными*, то есть действуют только в пределах тела функции в отличие от файла-сценария, все переменные которого *глобальные*.

Правила вывода комментариев те же, что и у файлов-сценариев.

Последняя конструкция **var=выражение** вводится, если требуется, чтобы функция возвращала результат вычислений. Если m-файл-функция завершается строкой с точкой с запятой (;), то для возврата значения функции используется программный оператор **return.**

Если *выходных параметров больше одного*, то структура модуля имеет вид:

function [var1,var2,…]=f_name(список параметров)
%Основной комментарий
%Дополнительный комментарий
Тело файла с любыми выражениями
var1=выражение
var2=выражение
……

Здесь **var1,var2,… –** имена переменных, которые являются выходными параметрами.

Такую функцию нельзя применять в математических выражениях, поскольку она возвращает не один результат. Данная функция используется (вызывается) как отдельный элемент программы в виде

[var1,var2,…]=f_name(список параметров).

Если такая функция используется в виде **f_name(список параметров)**, то возвращается значение только первого выходного параметра – переменной **var1**.

Если внутри функции целесообразно использовать глобальные переменные, то их нужно объявить с помощью команды

global var1 var2 …

Начиная с версии 5.0 в функции системы Matlab можно включать подфункции. Они имеют такую же структуру, как и основная функция, и записываются в теле основной функции.

3.2 Управляющие структуры языка программирования системы Matlab

3.2.1 Диалоговый ввод-вывод.

Команда **disp(X)** отображает массив, не печатая имя массива. Если X – строка, то отображается текст.

Пример

```
x=[1 2 3];
disp(x)
     1     2     3
disp('квадрат второго элемента=')
квадрат второго элемента=
disp(x(2)^2)
     4
```

R=INPUT('Сколько яблок?') даёт пользователю приглашение в текстовой строке и затем ожидает ввода с клавиатуры. Может быть введено любое Matlab-выражение, которое вычисляется с использованием переменных в текущей рабочей области, и результат возвращается в R. Если пользователь нажимает клавишу возврата каретки ENTER ничего не вводя, то вводится пустая матрица.

R=INPUT('Введите ваше имя','s') даёт приглашение в текстовой строке и ожидает ввода символьной строки. Напечатанный текст не вычисляется; символы просто возвращаются как Matlab-строка.

Пример

```
r=input('Введите угол в радианах')
введите угол в радианах 2*pi
r =
    6.2832
r=input('Введите ваше имя','s')
введите ваше имя 2*pi
r =
    2*pi
```

3.2.2 Циклы типа for-end.

Циклы типа **for-end** обычно используются для организации вычислений с заданным числом повторений цикла. Конструкция такого цикла имеет следующий вид:

for var=выражение
  Инструкция,…, Инструкция
end

Выражение чаще всего записывается как b:s:e, где b – начальное значение переменной цикла var, s – приращение (шаг) этой переменной и e – конечное значение управляющей переменной, при достижении которого цикл завершается. Возможна запись выражения в виде b:e, в этом случае s=1. Список выполняемых в цикле инструкций завершается оператором **end**.

Для досрочного выполнения цикла можно использовать оператор **break**. Как только этот оператор встречается в программе, цикл прерывается.

Возможно использование цикла в цикле.

Пример

```
for i=1:3
  for j=1:3
    a(i,j)=i+j;
  end
end
a
a =
    2     3     4
    3     4     5
    4     5     6
```

3.2.3 Циклы типа while-end.

Такие циклы имеют вид:

while Условие
  Инструкции
end

Цикл типа **while** выполняется до тех пор, пока выполняется Условие. Для прекращения выполнения цикла можно использовать оператор **break**.

Пример

```
x=1;i=1;
while x<=3
  y(i)=x;
  x=x+0.5;
  i=i+1;
end
y
y =
    1.0000    1.5000    2.0000    2.5000    3.0000
```

3.2.4 Условный оператор if-elseif-else-end.

Условный оператор **if** в общем виде записывается следующим образом:

if Условие
  Инструкции 1
elseif Условие
  Инструкции 2
else
  Инструкции 3
end

Эта конструкция допускает несколько частных вариантов. Простейший из них имеет вид:

if Условие
  Инструкции
end

Данный оператор работает следующим образом. Пока Условие возвращает логическое значение 1 (то есть выполняется), выполняются Инструкции. Оператор **end** указывает на конец списка Инструкций. Инструкции в списке разделяются запятыми или точками с запятыми. Если Условие возвращает логическое значение 0 (то есть не выполняется), то Инструкции также не выполняются.

Еще один вариант:

if Условие
  Инструкции 1
else
  Инструкции 2
end

В этом варианте выполняются Инструкции 1, если выполняется Условие 1, или Инструкции 2 – в противном случае.

Условие в операторе **if** записывается в виде:

Выражение_1 Оператор_отношения Выражение_2

В качестве Оператора_отношения используются следующие логические операторы: ==, <, >, <=, >=, ~=. Двойные символы не имеют между собой пробелов.

Пример

```
for i=1:3
  for j=1:3
    if i==j
      a(i,j)=2;
    elseif abs(i-j)==1
      a(i,j)=-1;
    else
      a(i,j)=0;
    end
  end
end
a
a =
     2    -1     0
    -1     2    -1
     0    -1     2
```

3.2.5 Переключатель switch-case-otherwise-end.

Для осуществления множественного выбора (или ветвления) используется конструкция с переключателем типа **switch:**

switch switch_Выражение
case case_Выражение
    Список_инструкций
case { case_Выражение1, case_Выражение2,… }
    Список_инструкций
…
otherwise,
    Список_инструкций
end

Выполняется первый оператор **case**, у которого case_Выражение соответствует switch_Выражению. Если ни одно из case_Выражений не соответствует switch_Выражению, то выполняется список инструкций после оператора **otherwise** (если он существует). Выполняется только один **case**, после чего выполнение продолжается с оператора после **end**.

Пример
Пусть существует m-файл-сценарий swit.m:

```
switch month
  case{1,2,3}
    disp('Первый квартал')
  case{4,5,6}
    disp('Второй квартал')
  case{7,8,9}
    disp('Третий квартал')
  case{10,11,12}
    disp('Четвертый квартал')
  otherwise,
    disp('Ошибка в данных')
end
```

Эта программа в ответ на значения переменной month (номер месяца) определяет номер квартала и выводит сообщение. Как это происходит, видно из следующей программы:

```
month=3;
swit
Первый квартал
month=10;
swit
Четвертый квартал
month=13;
swit
Ошибка в данных
```

3.2.6 Создание паузы в вычислениях.

Для остановки программы используется оператор **pause** в следующих формах:

pause – останавливает вычисления до нажатия любой клавиши;
pause(N) – останавливает вычисления на N секунд;
pause on – включает режим создания пауз;
pause off – выключает режим создания пауз.

4 Символьные операции в Matlab

4.1 Символьные операции

Символьными (или аналитическими) операциями называются такие операции, исходные данные на выполнение которых, а также результаты их выполнения определяются в виде символьных (формульных) выражений. В настоящее время имеется возможность выполнять символьные операции на компьютере. Для этого разработаны различные программные системы, такие как Reduce, Maple, Mathematica. Данные системы способны преобразовывать алгебраические выражения, находить аналитические решения систем линейных, нелинейных и дифференциальных уравнений, манипулировать полиномами, вычислять производные и интегралы, анализировать функции и находить их пределы и т. д. К символьным вычислениям относят также численные расчёты с произвольным числом цифр результатов и с отсутствующей погрешностью, поскольку это требует символьного представления чисел и особых алгоритмов выполнения операций с ними. Появление возможности выполнения символьных операций на компьютере привело к развитию нового научного направления – компьютерной математики (или компьютерной алгебры).

4.2 Выполнение символьных операций в Matlab

В систему Matlab входит пакет расширения Symbolic Math Toolbox (Symbolic), который базируется на ядре символьной математической системы Maple, лидирующей в области автоматизации аналитических решений. Система Matlab с пакетом Symbolic, включающим в себя чуть более сотни символьных команд и функций, намного уступает системе Maple по количеству таких команд и функций. В данный пакет включены наиболее важные и распространённые функции, так что возможности выполнения символьных операций в системе Matlab остаются весьма широкими. Помимо типовых аналитических вычислений (дифференцирование и интегрирование, упрощение математических выражений, подстановка и т. д.), пакет Symbolic позволяет реализовать арифметические операции с произвольной точностью.

4.3 Создание символьных переменных

Поскольку переменные системы Matlab по умолчанию не определены и задаются как векторные, матричные, числовые и т. д., то есть не имеющие отношения к символьной математике, то для реализации символьных вычислений

нужно, прежде всего, позаботиться о создании специальных символьных переменных. В простейшем случае их можно определить как строковые переменные, заключив имена в апострофы. Например, при вводе в окне управления команды

$$\sin('x')\text{^}2 + \cos('x')\text{^}2$$

и нажатии клавиши Enter получим результат ans=1.

Для создания символьных переменных или объектов используется функция **sym:**

x=sym('x') – возвращает символьную переменную с именем **'x'** и записывает результат в **x**;

x=sym('x','real') – возвращает символьную переменную вещественного типа с именем **'x'** и записывает результат в **x** (в общем случае символьные переменные рассматриваются как комплексные);

x=sym('x','unreal') – возвращает символьную переменную мнимого типа с именем **'x'** и записывает результат в **x**.

Возможно создание числа или матрицы в символьном виде с помощью записи вида **eps=sym('0.001')**.

4.4 Создание группы символьных переменных

Для создания группы символьных переменных или объектов используется функция **syms**:

syms x1 x2 … – создает группу символьных объектов, подобную выражениям **x1=sym('x1'); x2=sym('x2'); …**;

syms x1 x2 … real и **syms x1 x2 … unreal** – создают группы символьных объектов с вещественными (real) и невещественными (unreal) значениями. Последнюю функцию можно использовать для отмены задания вещественных объектов.

4.5 Создание списка символьных переменных

В математических выражениях могут использоваться как обычные, так и символьные переменные. Для выделения символьных переменных в некотором выражении применяется функция **findsym**:

findsym(s) – возвращает в алфавитном порядке список всех символьных переменных выражения **s**. При отсутствии таковых возвращается пустая строка;

findsym(s,n) – возвращает список **n** символьных переменных, ближайших к 'x' в алфавитном порядке в выражении **s**.

Пример

После выполнения m-файла-сценария

```
syms alpha x1 y b
a=1;
findsym(alpha+a+b)
```

```
findsym(cos(alpha)*b*x1 + 14*y,2)
findsym(y*(4+3*i) + 6*j)
```

будут выведены следующие результаты:

```
ans =
alpha, b
ans =
x1,y
ans =
y
```

Функция **findsym** позволяет упростить запись многих функций, поскольку она автоматически находит используемую в этих функциях символьную переменную.

4.6 Вывод символьного выражения

Система Matlab пока не способна выводить выражения и результаты их преобразований в естественной математической форме с использованием общепринятых спецзнаков для отображения интегралов, сумм, произведений и т. д. Тем не менее некоторые возможности близкого к математическому виду вывода обеспечивает функция **pretty**:

pretty(s) – даёт вывод выражения **s** в формате, приближённом к математическому;

pretty(s,n) – аналогична предшествующей функции, но задаёт вывод выражения **s** в **n** позициях строки (по умолчанию равно 79).

Функция **latex(s)** возвращает выражение **s** в форме текстового редактора LaTeX. Это позволяет использовать это выражение в LaTeX для получения выражения в его обычной математической форме.

Пример

После выполнения m-файла-сценария

```
syms x y
pretty(x^2/y^2)
z=latex(x^2/y^2)
```

будут выведены следующие результаты:

```
  2
 x
--
  2
 y

z =
\frac{x^2}{y^2}
```

4.7 Упрощение выражений

Функция **z = simplify(s)** поэлементно упрощает символьные выражения массива **s**. Если упрощение невозможно, то возвращается исходное выражение.

Пример

Программа

```
syms x
z=simplify(sin(x)^2+cos(x)^2)
```

возвращает результат z = 1.

Команда **simplify** в Symbolic не обладает в полной мере возможностями системы Maple по упрощению выражений. Дополнительные возможности обеспечивает функция **simple(s)**, которая выполняет различные упрощения для элементов массива **s** и выводит как промежуточные результаты, так и самый короткий конечный результат. При обращении к функции **simple** в форме

[R,HOW]=simple(s)

промежуточные результаты не выводятся. Конечные результаты упрощений содержатся в векторе **R**, а в строковом векторе **HOW** указывается выполняемое преобразование.

Пример

Программа

```
syms x
[R1,HOW1]=simple(cos(x)^2-sin(x)^2)
[R2,HOW2]=simple(2*cos(x)^2-sin(x)^2)
```

возвращает следующие результаты:

```
 R1 =
 cos(2*x)
 HOW1 =
combine(trig)
R2 =
3*cos(x)^2-1
HOW2=
simplify
```

Следует отметить, что в версии Matlab R2015a функция **simple** была удалена, вместо неё необходимо применять функцию **simplify**. При этом аналога записи «[R,HOW]=simple(s)» с использованием **simplify** не существует.

4.8 Вычисление производных

Для вычисления в символьном виде производных от выражения **s** служит функция **diff**:

diff(s) – возвращает символьное значение первой производной от символьного выражения или массива символьных выражений **s** по независимой переменной, определённой функцией **findsym**;

diff(s,n) – возвращает символьное значение **n**-й производной от символьного выражения или массива символьных выражений **s** по независимой переменной, определённой функцией **findsym**;

diff(s,v) – возвращает символьное значение первой производной от символьного выражения или массива выражений **s** по переменной **v**;

diff(s,v,n) или **diff(s,n,v)** – возвращает символьное значение **n**-й производной от символьного выражения или массива символьных выражений **s** по переменной **v**.

Пример

Программа

```
syms x t
y1=diff(sin(x^2))
y2=diff(t^6,6)
```

возвращает следующие результаты:

```
y1 =
2*cos(x^2)*x
y2 =
720
```

4.9 Вычисление интегралов

Для вычисления определённых и неопределённых интегралов в символьном виде служит функция **int**:

int(s) – возвращает символьное значение неопределённого интеграла от символьного выражения или массива символьных выражений **s** по переменной, которая автоматически определяется функцией **findsym**. Если **s** – константа, то вычисляется интеграл по переменной **'x'**;

int(s,a,b) – возвращает символьное значение определённого интеграла на отрезке интегрирования **[a,b]** от символьного выражения или массива символьных выражений **s** по переменной, которая автоматически определяется функцией **findsym**. Пределы интегрирования **a, b** могут быть как символьными, так и числовыми;

int(s,v) – возвращает символьное значение неопределённого интеграла от символьного выражения или массива выражений **s** по переменной **v**;

int(s,v,a,b) – возвращает символьное значение определённого интеграла от символьного выражения или массива символьных выражений **s** по переменной **v** с пределами интегрирования **[a,b]**.

Пример

Программа

```
syms x alpha u x1
y1=int(1/(1+x^2))
y2=int(sin(alpha*u),alpha)
y3=int(x1*log(1+x1),0,1)
```

возвращает следующие результаты:

```
y1 =
atan(x)
y2 =
-cos(alpha*u)/u
y3 =
1/4
```

4.10 Вычисление сумм рядов

Для аналитического вычисления суммы

$$R = \sum_{i=a}^{b} s(i),$$

где i – переменная суммирования; s(i) – общий член ряда; параметр b может быть конечным или бесконечным (inf), служит функция **symsum**:

symsum(s) – возвращает символьное значение суммы бесконечного ряда по переменной суммирования, найденной автоматически с помощью функции **findsym**;

symsum(s,a,b) – возвращает символьное значение суммы ряда по переменной суммирования, найденной автоматически с помощью функции **findsym**, при изменении этой переменной от **a** до **b**;

symsum(s,v) – возвращает символьное значение суммы бесконечного ряда по переменной суммирования **v**;

symsum(s,v,a,b) – возвращает символьное значение суммы ряда по переменной суммирования **v** при изменении этой переменной от **a** до **b**.

Пример

Программа

```
syms k n
y1=simplify(symsum(k,0,n-1))
y2=simplify(symsum(k,0,n))
y3=simplify(symsum(k^2,0,n))
```

возвращает следующие результаты:

```
y1 =
(n*(n - 1))/2
y2 =
(n*(n + 1))/2
y3 =
(n*(2*n + 1)*(n + 1))/6
```

4.11 Вычисление пределов

Для вычисления предела функции f(x) служит функция **limit**:

limit(f,a) – возвращает предел в точке **a** символьного выражения **f** по независимой переменной, найденной с помощью функции **findsym**;

limit(f) – возвращает предел в нуле символьного выражения **f** по независимой переменной, найденной с помощью функции **findsym**;

limit(f,x,a) – возвращает предел символьного выражения **f** в точке **x=a**;

limit(f,x,a,'right') – возвращает предел символьного выражения **f** в точке **x=a+0** (справа);

limit(f,x,a,'left') – возвращает предел символьного выражения **f** в точке **x=a-0** (слева).

Пример

После выполнения программы

```
syms x t a
y1=limit(sin(x)/x)
y2=limit((x-2)/(x^2-4),2)
y3=limit((1+2*t/x)^(3*x),x,inf)
y4=limit(1/x,x,0,'right')
v=[(1 + a/x)^x, exp(-x)];
y5=limit(v,x,inf,'left')
```

будет получен следующий результат:

```
y1 =
1
y2 =
1/4
y3 =
exp(6*t)
y4 =
inf
y5 =
[exp(a), 0]
```

4.12 Разложение функции в ряд Тейлора

Ряд Тейлора для функции $f(x)$ имеет вид:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n .$$

При $a = 0$ этот ряд называется рядом Маклорена. Для получения разложений аналитических функций в ряд Тейлора служит функция **taylor**:

taylor(f) – возвращает 6 членов ряда Маклорена;

taylor(f,n) – возвращает члены ряда Маклорена до (n-1)-го порядка;

taylor(f,a) – возвращает 6 членов ряда Тейлора в окрестности точки **a**;

taylor(f,a,n) – возвращает члены ряда Тейлора до (n-1)-го порядка в окрестности точки **a.**

Пример

После выполнения программы

```
syms x t
y1=taylor(exp(-x))
y2=taylor(log(x),6,1)
y3=taylor(sin(x),pi/2,6)
y4=taylor(x^t,3,t)
```

будет получен следующий результат:

```
y1 =
1-x+1/2*x^2-1/6*x^3+1/24*x^4-1/120*x^5
y2 =
x-1-1/2*(x-1)^2+1/3*(x-1)^3-1/4*(x-1)^4+1/5*(x-1)^5
```

```
y3 =
1-1/2*(x-1/2*pi)^2+1/24*(x-1/2*pi)^4
y4 =
1+log(x)*t+1/2*log(x)^2*t^2
```

4.13 Вычисление определителя матрицы, обращение матрицы

Функция **det(X)** вычисляет определитель (детерминант) квадратной матрицы X в символьном виде.

Для обращения (инвертирования) матрицы в символьном виде используется функция **inv(X)**.

Пример

После выполнения программы

```
syms a b c d t
A=[a b;c d];
y1=det(A)
B=[1/(2-t), 1/(3-t)
      1/(3-t), 1/(4-t)];
y2=inv(B)
```

будет получен следующий результат:

```
y1 =
a*d-b*c
y2 =
[     -(-3+t)^2*(-2+t), (-3+t)*(-2+t)*(-4+t)]
[ (-3+t)*(-2+t)*(-4+t),     -(-3+t)^2*(-4+t)]
```

Список литературы

1 **Дьяконов, В. П.** MATLAB. Полный самоучитель / В. П. Дьяконов. – Москва : ДМК Пресс, 2012. – 768 с.

2 **Герман-Галкин, С. Г.** Matlab & Simulink. Проектирование мехатронных систем на ПК / С. Г. Герман-Галкин. – Санкт-Петербург : КОРОНА-Век, 2008. – 368 с.

3 **Дьяконов, В.** Математические пакеты расширения MATLAB. Специальный справочник / В. Дьяконов, В. Круглов. – Санкт-Петербург : Питер, 2001. – 480 с.