

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

УТВЕРЖДАЮ
Декан ЭФФ

_____ Г.С. Евтушенко
« __ » _____ 2010 г.

В.В. Шестаков

Введение в «MatLab»

Методические указания к выполнению лабораторных работ
по курсу «Компьютерные технологии в приборостроении»
для студентов II курса, обучающихся по направлению 200100
«Приборостроение»

Издательство
Томского политехнического университета
2010

УДК 681.3.066(076.5)
ББК 32.973-018.1.я73
Ш514

Шестаков В.В.

Ш514 Введение в «MatLab»: методические указания к выполнению лабораторных работ по курсу «Компьютерные технологии в приборостроении» для студентов II курса, обучающихся по направлению 200100 «Приборостроение» / В.В. Шестаков; Национальный исследовательский Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2010. – 51 с.

УДК 681.3.066(076.5)
ББК 32.973-018.1.я73

Методические указания рассмотрены и рекомендованы
к изданию методическим семинаром кафедры
физических методов и приборов контроля качества ЭФФ
«__»_____ 2010 г.

Зав. кафедрой ФМПК
доктор технических наук

_____ *О.А. Сидуленко*

Председатель учебно-методической
комиссии

_____ *А.Н. Гормаков*

Рецензент

Кандидат технических наук, доцент кафедры информационно
измерительной техники ЭФФ ТПУ

В.В. Ширяев

© ГОУ ВПО «Национальный
исследовательский
Томский политехнический
университет», 2010
© Шестаков В.В., 2010

Основные теоретические сведения. Элементы языка программирования и визуализации расчетов в системе MATLAB.

Алфавит языка программирования

В MATLAB, как и в других системах, используются все буквы латинского алфавита от A до Z и арабские цифры от 0 до 9. Как и в C++, большие и малые буквы это разные переменные и константы. Кроме букв латинского алфавита используются все специальные символы клавиатуры компьютера.

Арифметические и логические операторы

Арифметические операторы в MATLAB состоят из матричных и арифметических операций. В таблице 1.1 приводится список арифметических операторов.

Таблица 1.1

Функция	Обознач.	(синтаксис)
Сложение	+	(M1+M2)
Вычитание	–	(M1–M2)
Матричное умножение	*	(M1*M2)
Поэлементное умножение массивов	.*	(M1.*M2)
Возведение матрицы в степень	^	(M1^x)
Поэлементное возведение массива в степень	.^	(M1.^x)
Деление матриц слева направо	/	(M1 / M2)
Поэлементное деление массивов слева направо	./	(M1 ./ M2)
Деление матриц справа налево	\	(M1 \ M2)
Поэлементное деление массивов справа налево	.\	(M1 .\ M2)
Транспонирование	‘	(M1 = M1’)

В математических выражениях операторы имеют определенный приоритет исполнения. В MATLAB приоритет логических операторов выше, чем арифметических, приоритет возведения в степень выше приоритетов умножения и деления, приоритет умножения и деления выше сложения и вычитания. Для повышения приоритета операций нужно использовать круглые скобки. Степень вложения скобок не ограничивается.

Операторы отношения служат для сравнения двух величин, векторов или матриц, все операторы отношения имеют две сравниваемые величины и записываются, как показано в таблице 1.2.

Функция	Оператор (синтаксис)
Равно	<code>== (x == y)</code>
Не равно	<code>~= (x ~= y)</code>
Меньше	<code>< (x < y)</code>
Больше	<code>> (x > y)</code>
Меньше или равно	<code><= (x <= y)</code>
Больше или равно	<code>>= (x >= y)</code>

Данные операторы выполняют поэлементное сравнение векторов или матриц одинакового размера и логическое выражение принимает значение 1 (True), если элементы идентичны, и значение 0 (False) в противном случае.

Логические операторы служат для реализации поэлементных логических операций над элементами одинаковых по размеру массивов согласно таблице 1.3.

Таблица 1.3

Функция	Оператор (синтаксис)
Логическое И	<code>&; and (and (a, b))</code>
Логическое ИЛИ	<code> ; or (or (a, b))</code>
Логическое НЕ	<code>~ ; not (not (a, b))</code>
Исключающее ИЛИ	<code>xor (xor (a, b))</code>
Верно, если все элементы вектора равны нулю	<code>any (any (a))</code>
Верно, если все элементы вектора не равны нулю	<code>all (all (a))</code>

Элементарные функции

Набор элементарных функций представим их описанием. В тригонометрических функциях углы измеряются в *радианах*.

Таблица 1.4

Функция	Синтаксис
$ x $ – модуль	<i>abs(x)</i>
e^x – экспонента	<i>exp(x)</i>
$\ln x$ – натуральный логарифм	<i>log(x)</i>
$\log_2 x$ – логарифм по основанию 2	<i>log2(x)</i>
$\lg x$ – десятичный логарифм	<i>log10(x)</i>
2^x – 2 в степени x	<i>pow(x)</i>
\sqrt{x} – квадратный корень	<i>sqrt(x)</i>
$\arccos x$ – арккосинус	<i>acos(x)</i>
$\operatorname{arcctg} x$ – арккотангенс	<i>acot(x)</i>
$\operatorname{arccosec} x$ – арккосеканс	<i>acsc(x)</i>
$\operatorname{arcces} x$ – арксеканс	<i>asec(x)</i>
$\operatorname{Arcsin} x$ – арксинус	<i>asin(x)</i>
$\operatorname{arctg} x$ – арктангенс	<i>atan(x)</i>
$\cos x$ – косинус	<i>cos(x)</i>
$\operatorname{ctg} x$ – котангенс	<i>cot(x)</i>
$\sec x$ – секанс	<i>sec(x)</i>
$\operatorname{cosec} x$ – косеканс	<i>csc(x)</i>
$\sin x$ – синус	<i>sin(x)</i>
$\operatorname{tg} x$ – тангенс	<i>tan(x)</i>
$\operatorname{arcch} x$ – арккосинус гиперболический	<i>acosh(x)</i>
$\operatorname{arccth} x$ – арккотангенс гиперболический	<i>acoth(x)</i>
$\operatorname{arccosech} x$ – арккосеканс гиперболический	<i>acsch(x)</i>

Функция	Синтаксис
$\operatorname{arcsech} x$ – арксеканс гиперболический	$\operatorname{asech}(x)$
$\operatorname{arcsh} x$ – арккосинус гиперболический	$\operatorname{asinh}(x)$
$\operatorname{arctgh} x$ – арктангенс гиперболический	$\operatorname{atanh}(x)$
$\operatorname{ch} x$ – косинус гиперболический	$\operatorname{cosh}(x)$
$\operatorname{ctgh} x$ – котангенс гиперболический	$\operatorname{coth}(x)$
$\operatorname{cosech} x$ – косеканс гиперболический	$\operatorname{csch}(x)$
$\operatorname{sech} x$ – секанс гиперболический	$\operatorname{sech}(x)$
$\operatorname{sh} x$ – синус гиперболический	$\operatorname{sinh}(x)$
$\operatorname{tgh} x$ – тангенс гиперболический	$\operatorname{tanh}(x)$

Следует помнить, что все элементарные функции должны записываться в программах *малыми буквами*. Существуют также специальные математические функции, на которых мы не будем останавливаться..

Понятие о файлах-сценариях и файлах-функциях

При загрузке системы MATLAB на мониторе появляется основное окно системы, в котором можно выделить *окно команд* (Command Window). Система готова к проведению вычислений и созданию программ в командном режиме. Для этого можно на языке MATLAB записывать программы. Операторы заканчиваются символом ; – точка с запятой. Одновременно точка с запятой *блокирует* вывод численного значения результата этого оператора в окне команд. В одной строке можно записать несколько операторов, а сами строки автоматически нумеруются при нажатии клавиши **Enter**. Если программа полностью записана и выходные величины не имеют символа ;, то после нажатия клавиши **Enter** она выполняется. Ниже программы появляется ее результат. В таком режиме выполнять решения задач нецелесообразно, т.к. исправить возможные ошибки после нажатия клавиши **Enter** уже нельзя. Поэтому записывать программы, их редактировать и отлаживать необходимо в так называемых М-файлах. М-файл создается при выполнении команды **New** меню **File**. Для ускорения этой команды выведена специальная пиктограмма в виде белой странички с загнутым уголком на панели инструментов. Щелкнув по пиктограмме стрелкой мышки, получаем окно М-файла, на котором можно записывать, редактировать и отлаживать любые программы решения научных и

инженерных задач. Данный М-файл по умолчанию имеет название **Untitled** (Безымянный). Чтобы дать ему имя, необходимо в меню этого окна **File** выполнить команду **Save as** и в другом окне указать папку и имя этого файла. После указания имени и сохранения М-файла он готов для выполнения записанной программы. Для этого необходимо щелкнуть мышкой по пиктограмме **Выполнить**. Она выполнена в виде страницы со стрелкой, направленной вниз ↓□. Результат выполнения программы или сообщения об ошибках появится в окне команд. Описанный процесс называется созданием *М-файла сценария сессии*. Файл-сценарий, именуемый также *Script*-файлом, имеет весьма простую структуру:

- % Основной комментарий, если необходимо.
- % Дополнительный комментарий, если необходимо.
- Тело программы с любыми выражениями.

Важными являются следующие свойства файлов-сценариев:

1. Они не имеют входных и выходных аргументов.
2. Работают с данными из рабочей области.
3. В процессе выполнения не компилируются.
4. Представляют собой *последовательность операций*, аналогичную той, что используется в сессии.

Кроме М-файла сценария, в MATLAB существует *М-файл функция*.

Отличие М-файла функции от сценария состоит в том, что он является аналогом подпрограммы типа *function* в языке Pascal.

Структура М-файла функции с одним выходным параметром имеет вид:

```
function var = f_name (Список параметров)
% Основной комментарий, если необходимо.
% Дополнительный комментарий, если необходимо.
Тело программы с любыми выражениями.
var = выражение
```

М-файл функция обладает такими свойствами:

1. Он начинается с ключевого слова function, после которого указывается имя переменной var – выходного параметра, имя самой функции f_name и список ее входных параметров, отделенных запятой. Внимание: Имя М-файла функции *должно совпадать* с самой f_name (именем самой функции). MATLAB автоматически присваивает данное имя при выполнении команды **Save as**.
2. Результат выполнения М-файла функции присваивается имени функции, которое может использоваться в математических выражениях подобно функциям *sin(x)*, *log(x)* и т. п.

3. Все переменные, используемые в файле-функции, являются локальными, т.е. действуют только в пределах тела функции.
4. Файл-функция является самостоятельным программным модулем, который связан с другими модулями и головной программой через входные и выходные параметры.
5. При обнаружении файла-функции он компилируется и затем выполняется.

Ниже в главах 2, 3 показаны примеры создания и использования в практических расчетах М-файлов сценариев и функций.

Основы программирования

Оператор присваивания

Программирование, т. е. создание определенного набора команд, в системе MATLAB является средством ее расширения и использования в решении специфических проблем. Отдельные вопросы программирования изложены выше, здесь рассмотрим правила, дополняющие синтаксис языка MATLAB.

Программы оперируют с *переменными* и *константами*. Переменные – это имеющие имена объекты, способные хранить разные по значению данные. В зависимости от этих данных переменные могут быть *числовыми* или *символьными*, *векторными* или *матричными*.

Для задания переменным определенных значений используется *оператор присваивания*, вводимый знаком равенства =

Имя _ переменной = Выражение ;

Типы переменных заранее *не декларируются*. Они определяются выражением, значение которого присваивается переменной. Имя переменной может содержать сколько угодно символов, но идентифицируется только 31 начальный символ. Имя любой переменной должно быть *уникальным*. Имя должно начинаться с *буквы*, может содержать буквы, цифры и символ подчеркивания *_*.

Недопустимо включать в имена *пробелы* и *специальные знаки*.

Перенос строки

Если математическое выражение выходит за размер экрана монитора, то целесообразно перенести его часть на следующую строку. Для этого используется символ многоточие ... – *три и более точки*. В командном режиме число возможных символов в одной строке – 4096, в М-файле – не ограничено, но с такими длинными строками работать неудобно. Поэтому применение в файлах-сценариях символа переноса строки улучшает наглядность программ.

Ввод и вывод данных

В языке MATLAB нет явных операторов ввода вывода данных. Эта проблема решается для ввода данных *оператором присваивания* и использованием *системных констант*. Вывод данных осуществляется еще проще. Для этого необходимо после математического выражения *не ставить символ ;* – точку с запятой. К системным константам относятся:

π	$= 3,1415 \dots$	–	число “ПИ”;
i или j		–	мнимые единицы;
NaN		–	неопределенность в виде $\frac{o}{o}$;
Inf		–	бесконечность типа $a/0$;
Ans			результат последней операции и др.

Форматы чисел

При вычислениях в MATLAB используется режим двойной точности. Однако, при выводе результатов, по умолчанию выдаются числа с 4 цифрами после десятичной точки в действительной форме. Чтобы изменить данную форму вывода, необходимо в программе перед выводимой величиной использовать команду **format name**, где **name** – имя формата. Для числовых данных **name** может быть следующим сообщением:

short – короткое представление в фиксированном формате (5 знаков);
short e – короткое представление в экспоненциальной форме (5 знаков мантииссы и 3 знака порядка);
long – длинное представление в фиксированном формате (15 знаков);
long e – длинное представление в экспоненциальной форме (15 знаков мантииссы и 3 знака порядка).

В качестве примера рассмотрим вывод вектора, содержащий 2 числа:

format name

$x = [5/3 \quad 1.2783 \ e - 7]$

В различных форматах вывод вектора x будет иметь следующий вид:

<i>short</i>	1.6667	0.0000
<i>short e</i>	1.6667E+000	1.2783E – 007
<i>long</i>	1.666666666666667	0.00000012783000
<i>long e</i>	1.666666666666667E+000	1.278300000000000E – 007

Задание формата сказывается только на форме вывода чисел.

Вычисления же происходят в режиме двойной точности, а ввод чисел осуществляется в любом удобном виде.

Формирование векторов и матриц

Описанные правила вычислений распространяются и на более сложные вычисления, которые при использовании обычных языков программирования (типа Pascal, Fortran, C++ и др.) требуют составления специальных программ. MATLAB специально предназначен для проведения сложных вычислений с векторами и матрицами. При этом по умолчанию предполагается, что каждая переменная – это вектор или матрица. Например, если задано $x = 1$, то это значит, что x – это вектор с одним элементом, равным 1. Если надо задать вектор из трех элементов, то их значения надо перечислить в квадратных скобках, разделяя пробелами.

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

В данном случае задан вектор-строка. Если разделить элементы точкой с запятой, то получим вектор-столбец.

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

Задание матрицы требует указания несколько строк. Для разграничения строк используется символ ; (точка с запятой).

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

Для указания отдельного элемента вектора или матрицы используются выражения вида $V(i)$ или $T(i, j)$. Например:

```
>>T(3,2)
      ans    =
          8
```

Если элементу $T(i, j)$ нужно присвоить новое значение x , то используют оператор присваивания

$T(3,2) = x$;

Выражение $T(i)$ с одним индексом дает доступ к элементам матрицы, развернутым *в один столбец*. Такая матрица образуется из исходной, если подряд выписать ее столбцы. Например:

```
>>T (3)
ans      =
          7

>>T (8)
ans      =
          6
```

Наряду с операциями над отдельными элементами матриц и векторов MATLAB позволяет производить арифметические операции сразу над всеми элементами. Для этого перед знаком операции *ставится точка*. Имеются также ряд особых функций для задания векторов и матриц. Отметим функции *ones* и *zeros*. Эти функции служат для создания одномерных и многомерных массивов. Функция *ones* создает массив с единичными элементами

```
>> a = ones (3, 2)
a      =
          1   1
          1   1
          1   1
```

Функция *zeros* создает массив с нулевыми элементами

```
>> b = zeros (2, 3)
b      =
          0   0   0
          0   0   0
```

В краевых задачах поиска частот собственных колебаний и критических сил потери устойчивости упругих систем функция *zeros* находит широкое применение, что существенно упрощает тексты соответствующих программ.

Типы данных

В MATLAB определены типы данных, представляющие собой многомерные массивы:

1. *single* – числовые массивы с числами одинарной точности;
2. *double* – числовые массивы с числами удвоенной точности;
3. *char* – строчные массивы с элементами – символами;
4. *sparse* – разреженные матрицы с элементами – числами двойной точности;
5. *cell* – массивы ячеек;

6. *structure* – массивы структур с полями;
7. *function – handle* – дескрипторы функций;
8. *int8*, ..., *uint32* – массивы 8-, 16-, 32-разрядных целых чисел со знаками и без знаков.

Оператор двоеточие :

Весьма часто необходимо выполнить формирование упорядоченных числовых последовательностей. Такие последовательности нужны для создания векторов или значений аргументов x при построении графиков. В MATLAB для этого используется оператор двоеточие :, который представляется следующим образом:

$x = \text{Начальное_значение} : \text{Шаг} : \text{Конечное_значение} ;$

Эта конструкция создает возрастающую последовательность чисел, которая начинается с начального значения, изменяется на заданный шаг и завершается конечным значением. Если шаг не задан, то он принимает значение 1. Если конечное значение указано меньшим, чем начальное значение, – то выдается сообщение об ошибке. Примеры:

```
>> x      = 0 : 5
      x
          0  1  2  3  4  5

>> cos(x)
      ans
          1.0000  0.5403 -0.4161 -0.9900 -0.6536  0.2837

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

и т.д.

Оператор разветвления if

Условный оператор **if** в MATLAB записывается в общем виде так:

***if* Логическое условие Оператор 1 *elseif* Логическое условие Оператор 2 *else* Оператор 3 *end* ;**

Эта конструкция имеет несколько частных вариантов:

***if* Логическое условие Оператор 1 *end* ;**

***if* Логическое условие Оператор 1 *else* Оператор 2 *end* ;**

Логическое условие записывается в виде:

Выражение 1 Оператор отношения Выражение 2

В качестве *операторов отношения* используются операторы: $=$, $<$, $>$, \leq , \geq , \sim . Если логическое условие принимает значение 1 (*true* – истина), то выполняются соответствующие операторы. Если логическое условие принимает значение 0 (*false* – ложь), то операторы, следующие за логическим условием, не выполняются. Оператор *end* указывает на конец условного оператора *if*. В понятие *Оператор 1* входят один или несколько операторов. В последнем случае они разделяются символами $,$ (запятой) или $;$ (точкой с запятой).

Как и в других алгоритмических языках, оператор *if* позволяет осуществить разветвление процесса вычислений в зависимости от какого-либо условия. Примеры применения этого оператора представлены в главе 3.

Операторы циклов

В MATLAB существует 3 типа операторов цикла. С оператором : (двоеточие) мы познакомились в п.1.5.7. Следующий оператор *for ... end* используется для организации цикла с фиксированным числом повторений. Он имеет вид:

<i>for var</i> = Выражение Операторы <i>end</i> ;
--

Здесь *var* – счетчик цикла – любая переменная, обычно это *i*, *j*, *k*, *l*, *m* и т. д. *Выражение* записывается в виде $s : d : e$, где s – начальное значение счетчика цикла *var*, d – шаг изменения и e – конечное значение *var*.

Возможна и запись в виде $s : e$, тогда $d = 1$. Список операторов завершается ключевым словом *end*. Оператор *continue* передает управление в следующую итерацию цикла, пропуская операции, которые записаны за ним. Оператор *break* используется для досрочного прерывания цикла. Возможны вложенные циклы

\gg *for i* = 1 : 3 *for j* = 1 : 3 *a* (*i*,*j*) = *i* * *j* ; *end* ; *end* ;

В результате выполнения этого цикла формируется матрица *a*

```
 $\gg$  a
    a      =
         1   2   3
         2   4   6
         3   6   9
```

Циклы типа *while ... end* выполняются до тех пор, пока выполняется заданное условие. Оператор записывается в виде:

<i>while</i> Логическое условие Операторы <i>end</i> ;

Сообщения об ошибках и исправление ошибок

Система MATLAB контролирует правильность написания программ и, при наличии ошибок, выдает соответствующее сообщение в окне команд. При этом указывается номер строки, где допущена ошибка, и характер ошибки. После уяснения сути ошибки ее необходимо исправить в тексте программы, запомнить M-файл командой **Save** и снова выполнить программу. Перед этим желательно очистить окно команд от сообщения об ошибках (чтобы не загромождать полученную картинку) с помощью команды **Clear Command Windows** (Очистить окно команд) в меню *Edit*.

Вычисление определителя квадратной матрицы

Для вычисления определителя квадратной матрицы используется функция $\det(a)$. Если матрица a содержит только целые числа, то результат – тоже целое число. Определитель вычисляется на основе треугольного разложения методом исключения Гаусса. Пример:

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

>> det(a)
ans       =
          - 29
```

Данная функция широко используется в задачах поиска спектров частот собственных колебаний и критических сил потери устойчивости упругих систем (см. главу 3).

Обращение матриц

Обращение матриц – одна из наиболее распространенных операций. *Обратной* называют матрицу, получаемую в результате деления единичной матрицы E на исходную матрицу x , т. е. $x^{-1} = E/x$. Эту процедуру выполняет функция $\text{inv}(x)$, которая вычисляет элементы обратной матрицы для исходной квадратной матрицы x . Выдается предупреждающее сообщение, если матрица x плохо масштабирована или близка к вырожденной. На практике вычисление обратной матрицы не так уж необходимо. Чаще обращение применяют для решения систем линейных алгебраических уравнений вида $ax = b$. Один из путей решения этой системы – $x = \text{inv}(a) * b$, хотя лучше использовать метод

исключения Гаусса без формирования обратной матрицы, например $x = a/b$ или $x = b/a$.

Основы графической визуализации вычислений

Во многих областях науки и техники численное решение задач недостаточно для анализа результатов. Необходима еще графическая интерпретация в виде эпюр параметров напряженно-деформированного состояния элементов упругих систем, формы колебаний и потери устойчивости, поведение решений на заданном интервале и т. п. MATLAB позволяет решать эти задачи достаточно простыми процедурами. Вначале необходимо задать интервал изменения аргумента x от начального значения x_0 до конечного x_k с шагом Δx , что осуществляется оператором двоеточие: $x_0 : \Delta x : x_k$. Далее используется команда построения графика какой-либо функции $y = f(x)$, которая носит имя *plot*.

Plot строит не истинный график функции $f(x)$, а лишь заданное числом элементов вектора x число точек. Эти точки затем соединяются отрезками прямых, т. е. выполняется кусочно - линейная интерполяция данных графика. Если число точек достаточно велико, то полученная кривая воспринимается как вполне истинный график функции $y = f(x)$, при 10 – 20 точках получается ломаная кривая.

Построение графиков отрезками прямых

Для построения графика функции $y = f(x)$ необходимо задать совокупность точек x и y . Для аргумента x это выполняется оператором двоеточие, для y – *надлежащим программированием* выражения для функции, т. е. необходимо применить знаки арифметических операций над массивами:

$:\cdot * ; \cdot / ; \cdot ^$.

Для отображения таких функций используется декартова прямоугольная система координат. Команда построения графика функции $y = f(x)$ *plot* имеет ряд параметров, которые рассмотрим ниже. *plot(x, y)* – строит график функции $y = f(x)$, координаты точек (x, y) которой берутся из векторов *одинакового размера* x, y .

plot(x, y, s) – аналогична команде *plot(x, y)*, но тип линии графика можно задавать с помощью строковой константы s , значения которой представлены в таблице 1.5.

Таблица 1.5

Цвет линии	Тип точки	Тип линии
$s = y$ – желтый	$s = \bullet$ – точка	$s = -$ – сплошная

<i>m</i> – фиолетовый <i>c</i> – голубой <i>r</i> – красный <i>g</i> – зеленый <i>b</i> – синий <i>w</i> – белый <i>k</i> – черный	O – окружность x – крест + – плюс * – звездочка s – квадрат d – ромб v – треугольник (вниз) ^ – треугольник (вверх) < – треугольник (влево) > – треугольник (вправо) p – пятиугольник h – шестиугольник	: – двойной пунктир - . - штрихпунктир - - - штриховая
--	--	--

Таким образом, с помощью строковой константы *s* можно менять цвет линии, представлять узловые точки различными отметками и менять тип линии графика. Рассмотрим пример построения графиков трех функций:

$y_1 = \sin x$; $y_2 = \sin^2 x$; $y_3 = \sin^3 x$ с различным стилем:

```
>> x = - 2 * pi : 0 . 01 * pi : 2 * pi ; y1 = sin (x) ; y2 = sin (x) . ^ 2 ; y3 = sin (x) . ^ 3 ;
```

```
>> plot(x, y1, ' - m ', x, y2, ' - . + r ', x, y3, ' - - ok ')
```

Здесь график функции y_1 строится *сплошной фиолетовой линией*, график y_2 строится *штрихпунктирной линией с точками в виде знака “+” красного цвета*, график y_3 строится *штриховой линией с кружками черного цвета*. (цвет линии, тип точки, и тип линии в константе *s* можно задавать в любой последовательности)

Создание массивов данных для трехмерной графики

Трехмерные поверхности описываются функцией двух переменных вида: $z = f(x, y)$. Построение трехмерных графиков требует определение для x и y двумерных массивов – матриц. Для создания таких массивов служит функция *meshgrid*, которая записывается следующим образом:

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

В основном она используется совместно с функциями построения графиков трехмерных поверхностей. Функция преобразует область заданную векторами x и y , в массивы X и Y , которые могут быть использованы для вычисления функции двух переменных и построения трехмерных графиков. Строки выходного массива X являются копиями вектора x , а столбцы Y – копиями вектора y .

Пример:

```
>> [X Y] = mesh grid (4 : 7 , 9 : 13)
```



```

X      =
      4   5   6   7
      4   5   6   7
      4   5   6   7
      4   5   6   7
      4   5   6   7

Y      =
      9   9   9   9
     10  10  10  10
     11  11  11  11
     12  12  12  12
     13  13  13  13

```

Приведем еще пример применения функции *meshgrid*:

```
>> [X Y] = meshgrid(-1 : 0.1 : 1, -1 : 0.1 : 1);
```

Такой вызов функции создает опорную плоскость для построения трехмерной поверхности при изменении x и y от -1 до 1 с шагом 0.1 .

Построение графиков поверхностей

Для построения графиков функции $z = f(x, y)$ используется команды *plot3* (...), которая является аналогом команды *plot* (...). Она строит аксонометрическое изображение трехмерной поверхности и имеет следующие формы:

plot3 (x, y, z) – строит массив точек, представленных векторами x, y, z и соединяет их отрезками прямых.

plot3 (X, Y, Z), где X, Y, Z – три матрицы одинакового размера, строит точки с координатами $X(i, :)$, $Y(i, :)$ и $Z(i, :)$ и соединяет их отрезками прямых. Пример построения графика трехмерной поверхности $Z = x^2 + y^2$:

```

>> [X Y]          = meshgrid([-3 : 0.15 : 3])
>> Z              = X.^2 + Y.^2;
>> plot3(X, Y, Z)

```

plot3 (X, Y, Z, S) – обеспечивает построение графика поверхности, но со спецификацией стиля линий и точек, соответствующей спецификации команды *plot*.

Включение и выключение масштабной сетки

При построении графиков наряду с разметкой осей часто необходимо иметь масштабную сетку. Команды *grid* позволяют управлять этим

процессом. Если после команды построения графиков добавить команду включения или выключения масштабной сетки, то можно получить график с требуемым видом:

grid on – добавляет сетку к текущему графику;

grid off – отключает сетку;

grid – последовательно производит включение и отключение сетки.

Представление нескольких графиков в одном окне

Иногда необходимо в одном окне поместить несколько графиков без наложения их друг на друга. Особенно удобно такое представление при построении эпюр напряженно-деформированного состояния элементов стержневых и пластинчатых систем. Для этого служит команда *subplot*, которую необходимо записать перед командой *plot*.

subplot (m, n, p) – разбивает графическое окно на $m \times n$ подокон, при этом m – число подокон по горизонтали, n – число окон по вертикали, p – номер подокна, в которое будет выводиться текущий график.

Ввод текста на график с помощью мыши

Для маркировки графиков можно ввести любой текст с помощью мыши командой *gtext*. Команда помещается после команды *plot*.

gtext ('string') – выводит на график перемещаемый мышкой маркер в виде крестика. Установив маркер в нужное место и щелкнув кнопкой мыши, получим текст на графике.

Управление свойствами осей графиков

Если не задавать масштаб графика, то он строится командой *plot* автоматически. Не всегда этот масштаб удовлетворяет пользователя. Команда *axis* позволяет установить любой масштаб.

axis ([x min x max y min y max]) – устанавливает нужный диапазон координат графика по осям x и y .

В заключение отметим, что более полно о возможностях графической визуализации MATLAB можно узнать в специализированных изданиях [1] и др.

Лабораторная работа №1

MATLAB в задачах вычислительной математики

Цель работы:

Изучение приемов работы с командным окном ('Command window') системы MatLab. Использование файлов сценариев и файлов – функций для решения простейших задач вычислительной математики и визуализации результатов вычислений.

Общие сведения:

Программы, реализующие какой-либо численный метод, необходимо записывать в М-файл. Если не дать имени М-файлу, то он запишется при выполнении программы в рабочую папку под именем Untitled (Безымянный). Такой ситуации следует избегать для исключения появления множества файлов с неопределенным именем. Рассмотрим решение различных проблем вычислительной математики, имеющих важное значение при изучении различных наук.

Табулирование функций

Данная задача широко используется в экологии, теплофизике и других дисциплинах. Обычно функции, описывающие какой-либо процесс, весьма громоздки и создание таблиц их значений требует большого объема вычислений.

Рассмотрим два случая табулирования функции:

1. С постоянным шагом изменения аргументов.
2. С произвольным набором значений аргумента.

Алгоритм реализуется путем организации какого-либо цикла.

Пример 1. Вычислить

$$y_i = R \sqrt[3]{\left| \ln(1 - \lambda^{x_i})^2 - x_i^3 \right|},$$

при $R = 4.28 \cdot 10^{-2}$; $\lambda = 2.87$;

x_i изменяется с шагом $\Delta x = 2$; $x_n = 2$; $x_k = 10$.

Введем обозначение $\lambda \rightarrow la = 2.87$.

Протокол программы:

$R = 4.28e-02$; $la = 2.87$;

% Задается начальное значение x , шаг dx и конечное значение x
 $x = 2.0 : 2.0 : 10.0;$

$$y = R_*(abs(log((1 - la.^x).^2) - x.^3)).^(1/3); \quad [x; y]$$

% Для вывода значения y в конце строки символ ; *не ставится!*

В окне команд появляются после нажатия кнопки *выполнить* значения функции y , которые затем можно скопировать в какой-либо файл.

Результаты вычислений:

ans =

2.0000	4.0000	6.0000	8.0000	10.0000
0.0682	0.1634	0.2517	0.3386	0.4250

Пример 2. Вычислить и вывести на экран значения функции

$$y_i = \frac{1 + \sin^2(b^2 + x_i^2)}{\sqrt[3]{a^2 + x_i^2}};$$

при $x_1 = 12.8; x_2 = 23.4; x_3 = 27.2; x_4 = 17.8; x_5 = 16.3; x_6 = 14.9; a = 1.35; b = 0.98.$

Данную задачу можно программировать не изменяя обозначения переменных. Цикл организуется для одномерного массива.

Протокол программы:

$a = 1.35; b = 0.98; x(1) = 12.8; x(2) = 23.4; x(3) = 27.2; x(4) = 17.8; x(5) = 16.3; x(6) = 14.9;$

$for \quad m = 1 : 6 \quad y = (1 + \sin(b^2 + x(m).^2).^2)/(a^2 + x(m).^2).^(1/3), end;$

% В конце строки вычисления функции y символ ; *не ставится.*

y	=	0.3609
y	=	0.2327
y	=	0.1473
y	=	0.1800

$$y = 0.1771$$

$$y = 0.1658$$

Данные вычислений можно вывести в виде таблицы, если использовать запись $[x; y]$ без точки с запятой или $[x \quad y]$.

Варианты заданий

Составить программу вычисления значений функции y_i для значений аргумента x_i . Данные взять из таблицы 2.1.

Таблица 2.1

№ п/п	Функция $y_i = f(x_i)$	Задача А					Задача В				
		a	b	x_H	x_K	Δx	x_1	x_2	x_3	x_4	x_5
1	$y = \frac{1 + \sin^2(b^3 + x^3)}{\sqrt[3]{b^3 + x^3}}$	-	2.5	1.28	3.28	0.4	1.1	2.4	3.6	1.7	3.9
2	$y = \frac{\sqrt[3]{ax + b}}{\lg^2 x}$	1.35	0.98	1.14	4.24	0.62	0.35	1.28	3.51	5.21	4.16
3	$y = \frac{1 + \lg^2 \frac{x}{a}}{b - e^{\frac{x}{a}}}$	2.0	0.95	1.25	2.75	0.3	2.2	3.78	4.51	6.58	1.2
4	$y = \sqrt[4]{x^2 - 2,5} + \sqrt[3]{\lg x^2}$	-	-	1.25	3.25	0.4	1.84	2.71	3.81	4.56	5.62
5	$\begin{cases} y = \frac{\lg^2(a^2 + x)}{(a + x)^2}, & x > 5 \\ \frac{(a + bx)^{2,5}}{1,8 + \cos^3(ax)}, & \geq 5 \end{cases}$	-2.5	3.4	3.5	6.5	0.6	2.89	3.54	5.21	6.28	3.48
6	$y = 1,2^x - x^{1,2}, \quad x \geq 1$ $\arccos x, \quad x < 1$	-	-	0.2	2.2	0.4	0.1	0.9	1.2	1.5	2.3
7	$y = \frac{a^x - b^x}{\lg \frac{a}{b}} \sqrt[3]{ab}$	0.4	0.8	3.2	6.2	0.6	4.48	3.56	2.78	5.28	3.21
8	$y = \frac{b^3 + \sin^2 ax}{\arccos(xbx) + e^{-x/2}}$	1.2	0.48	0.7	2.2	0.3	0.25	0.36	0.56	0.94	1.28
9	$y = \frac{\lg(x^2 - 1)}{\log_5(ax^2 - b)}$	1.1	0.09	1.2	2.2	0.2	1.21	1.76	2.53	3.48	4.52

№	Функция	Задача А					Задача В				
10	$y = \frac{\arccos(x^2 - b^2)}{\arcsin(x^2 - a^2)}$	0.05	0.06	0.2	0.95	0.15	0.15	0.26	0.37	0.48	0.56
11	$y = \arcsin(x^a) + \arccos(x^b)$	2.0	3.0	0.11	0.36	0.05	0.08	0.26	0.35	0.41	0.53
12	$y = a^{x^2-1} - \lg(x^2 - 1) + \sqrt[3]{x^2 - 1}$	1.6	-	1.2	3.7	0.5	1.28	1.36	2.47	3.68	4.56
13	$y = \frac{a\sqrt{x} - b\log_5 x}{\lg x-1 }$	4.1	2.7	1.2	5.2	0.8	1.9	2.15	2.34	2.73	3.16
14	$y = \sqrt{\frac{ a-bx }{\lg^3 x}}$	7.2	4.2	1.81	5.31	0.7	2.4	2.8	3.9	4.7	3.16
15	$y = (\arcsin^2 x + \arccos^4 x)^3$	-	-	0.26	0.66	0.08	0.1	0.35	0.4	0.55	0.6
16	$y = \frac{\ln_a b^2 - x^2 }{\sqrt[5]{ x^2 - a^2 }}$	2.0	1.1	0.08	1.08	0.2	0.1	0.3	0.4	0.45	0.65
17	$y = \frac{a + tg^2 bx}{b + ctg^2 ax}$	0.1	0.5	0.15	1.37	0.25	0.2	0.3	0.44	0.6	0.56
18	$y = \frac{(a + bx)^{2.5}}{1 + \lg(a + bx)}$	2.5	4.6	1.1	3.6	0.5	1.2	1.28	1.36	1.46	2.35
19	$y = \frac{\lg^2(a + x)}{(a + x)^2}$	2.0	-	1.2	4.2	0.6	1.16	1.32	1.47	1.65	1.93
20	$y = \frac{\sqrt[3]{(x-a)^2} + \sqrt[5]{ x+b }}{\sqrt[9]{x^2 - (a+b)^2}}$	0.8	0.4	1.23	7.23	1.2	1.88	2.26	3.84	4.55	-6.21
21	$y = (\sin^3 x + \cos^3 x) \ln x$	-	-	0.11	0.36	0.05	0.2	0.3	0.38	0.43	0.57
22	$y = a^{x^2-1} - \lg(x^2 - 1) + \sqrt[3]{x^2 - 1}$	2.25	-	1.2	2.7	0.3	1.31	1.39	1.44	1.56	1.92
23	$y = \frac{a\sqrt[3]{x} - b\log_5 x}{\lg^3(x-1)}$	4.1	2.7	1.5	3.5	0.4	1.9	2.15	2.34	2.74	3.16
24	$y = \sqrt[5]{\frac{a+bx}{\lg^3 x}}$	7.2	1.3	1.56	4.71	0.63	2.4	2.8	3.9	4.7	3.16
25	$y = \sqrt[7]{\arcsin^4 x + \arccos^6 x}$	-	-	0.22	0.92	0.14	0.1	0.35	0.4	0.55	0.6
26	$y = \frac{\log_a(b^2 - x^2)}{\sqrt[3]{ x^2 - a^2 }}$	2.0	4.1	0.77	1.77	0.2	1.24	1.38	2.38	3.21	0.68
27	$y = \frac{\sqrt[3]{a} + tg^{4.5} bx}{\sqrt[5]{b} + ctg^{2.7} ax}$	0.1	0.5	0.33	1.23	0.18	0.5	0.36	0.40	0.62	0.78
28	$y = \frac{\sin(a + bx)^{3.5}}{1 + \cos[\lg(a + bx)]}$	2.5	4.6	1.15	3.05	0.38	1.2	1.36	1.57	1.93	2.25

№	Функция	Задача А					Задача В				
29	$y = \frac{\lg[\lg^3(a+x)]}{\sqrt[3]{(a+x)^2}}$	2.0	-	1.08	1.88	0.16	1.16	1.35	1.48	1.52	1.96
30	$y = \frac{\sqrt[3]{x-a} + \sqrt[5]{x+b}}{\sqrt[3]{x} - \sqrt[9]{x^2 - (a+b)^2}}$	0.8	0.4	1.42	3.62	0.44	1.6	1.81	2.24	2.65	3.38

Решение систем линейных алгебраических уравнений методом исключения Гаусса

К решению систем линейных уравнений сводятся многочисленные практические задачи, например различные краевые задачи для обыкновенных и в частных производных дифференциальных уравнений. Можно с полным основанием утверждать, что данная проблема является одной из самых распространенных и важных задач вычислительной математики.

Пусть задана система n линейных алгебраических уравнений с n неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \text{-----} \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases} \quad (2.1)$$

Система уравнений (2.1) в матричной форме представляется следующим образом:

$$\mathbf{AX} = \mathbf{B}, \quad (2.2)$$

где \mathbf{A} – квадратная матрица коэффициентов, размером $n \times n$ строк и столбцов;

\mathbf{X} – вектор-столбец неизвестных;

\mathbf{B} – вектор-столбец правых частей.

Систему уравнений (2.2) можно решить различными методами. Один из наиболее простых и эффективных методов является метод исключения Гаусса и его модификации. Алгоритм метода основан на приведении матрицы \mathbf{A} к треугольному виду (прямой ход) и последовательном вычислении неизвестных (обратный ход). Эти процедуры можно

выполнять над невыраженными матрицами, в противном случае метод Гаусса неприменим.

Недостатком метода является накапливание погрешностей в процессе округления, поэтому метод Гаусса без выбора главных элементов используется обычно для решения сравнительно небольших ($n \leq 100$) систем уравнений с плотно заполненной матрицей и не близким к нулю определителем. Если матрица **A** сильно разрежена, а ее определитель не близок к нулю, то метод Гаусса пригоден для решения больших систем уравнений. В MATLAB имеется обширный арсенал методов решения систем уравнений (2.2) методом исключения Гаусса. Для этого применяются следующие операторы

$\boxed{\quad / \quad}$ – правое деление;

$\boxed{\quad \backslash \quad}$ – левое деление;

$\boxed{\quad \wedge - 1 \quad}$ – возведение в степень -1 ;

$\boxed{\text{inv}(\mathbf{A})}$ – обращение матрицы **A**.

Выражения

$$\mathbf{X} = \mathbf{B}/\mathbf{A}$$

$$\mathbf{X} = \mathbf{B} * \mathbf{A}^{\wedge - 1}$$

$$\mathbf{X} = \mathbf{B} * \text{inv}(\mathbf{A})$$

$$\mathbf{X} = \mathbf{A} \backslash \mathbf{B}$$

дают решения ряда систем линейных уравнений $\mathbf{AX} = \mathbf{B}$, где **A** – матрица размером $m \times n$, **B** – матрица размером $n \times 1$.

Пример 3.

Решить систему 4-х линейных уравнений:

$$\begin{cases} 1.1161x_1 + 0.1397x_2 + 0.1254x_3 + 0.1490x_4 = 1.5471; \\ 0.1582x_1 + 0.1768x_2 + 1.1675x_3 + 0.1871x_4 = 1.6471; \\ 0.1968x_1 + 1.2168x_2 + 0.2071x_3 + 0.2271x_4 = 1.7471; \\ 0.2368x_1 + 0.2568x_2 + 0.2471x_3 + 1.2671x_4 = 1.8471. \end{cases}$$

Протокол программы (в М-файле)

$$\begin{aligned} a &= \begin{bmatrix} 1.1161 & 0.1397 & 0.1254 & 0.1490 \\ 0.1582 & 0.1768 & 1.1675 & 0.1871 \\ 0.1968 & 1.2168 & 0.2071 & 0.2271 \\ 0.2368 & 0.2568 & 0.2471 & 1.2671 \end{bmatrix}; \\ b &= [1.5471 ; 1.6471 ; 1.7471 ; 1.8471]; \end{aligned}$$

$$X4 = a \setminus b$$

Эта программа выдает решение заданной системы с помощью четвертого оператора в виде матрицы – столбца

$$\begin{aligned} X4 &= \\ &1.0406 \\ &0.9351 \\ &0.9870 \\ &0.8813 \end{aligned}$$

Внимание. В М-файле матрица ***a*** набирается по строкам, а элементы матрицы правых частей ***b*** отделяются символом ; , т. е. тоже набираются по строкам. Решение другими операторами системы уравнений (2.2) требует набора матрицы ***a*** по столбцам, а элементы правых частей ***b*** отделяются только пробелом!

$$\begin{aligned} a &= \begin{bmatrix} 1.1161 & 0.1582 & 0.1968 & 0.2368 \\ 0.1397 & 0.1768 & 1.2168 & 0.2568 \\ 0.1254 & 1.1675 & 0.2071 & 0.2471 \\ 0.1490 & 0.1871 & 0.2271 & 1.2671 \end{bmatrix}; \\ b &= [1.5471 \quad 1.6471 \quad 1.7471 \quad 1.8471]; \end{aligned}$$

$$X1 = b/a$$

$$X2 = b * a^{-1}$$

$$X3 = b * inv(a)$$

Результаты решения

$$\begin{aligned} X1 &= \\ &1.0406 \quad 0.9351 \quad 0.9870 \quad 0.8813 \\ X2 &= \\ &1.0406 \quad 0.9351 \quad 0.9870 \quad 0.8813 \\ X3 &= \end{aligned}$$

1.0406 0.9351 0.9870 0.8813

Варианты заданий. Решить систему линейных алгебраических уравнений с помощью 4-х операторов. Данные взять из таблицы 2.2.

Таблица 2.2

1	$a_{ij} = \begin{bmatrix} 2 & -4 & 3 & 1 \\ -1 & 5 & -7 & -3 \\ 10 & -2 & 4 & 4 \\ -1 & 1 & -1 & -1 \end{bmatrix}; b_i = \begin{bmatrix} 7 \\ -24 \\ 34 \\ -6 \end{bmatrix}$	2	$a_{ij} = \begin{bmatrix} 5 & -3 & 4 & -2 \\ 10 & 3 & -4 & 2 \\ 7 & -5 & 8 & -10 \\ 4 & 5 & -8 & 10 \end{bmatrix}; b_i = \begin{bmatrix} 4 \\ 11 \\ 0 \\ 11 \end{bmatrix}$
3	$a_{ij} = \begin{bmatrix} 2 & 2 & -1 & 1 \\ 4 & 3 & -1 & 2 \\ 8 & 5 & -3 & 4 \\ 3 & 3 & -2 & 2 \end{bmatrix}; b_i = \begin{bmatrix} 4 \\ 6 \\ 12 \\ 6 \end{bmatrix}$	4	$a_{ij} = \begin{bmatrix} 1 & 1 & -6 & 4 \\ 3 & -1 & -6 & -4 \\ 2 & 3 & 9 & 2 \\ 3 & 2 & 3 & 8 \end{bmatrix}; b_i = \begin{bmatrix} 6 \\ 2 \\ 6 \\ -7 \end{bmatrix}$
5	$a_{ij} = \begin{bmatrix} 1 & 5 & 3 & 4 \\ 7 & 14 & 20 & 27 \\ 5 & 10 & 16 & 19 \\ 3 & 5 & 6 & 13 \end{bmatrix}; b_i = \begin{bmatrix} 0 \\ 0 \\ -2 \\ 5 \end{bmatrix}$	6	$a_{ij} = \begin{bmatrix} 2 & 4 & 3 & 4 \\ 4 & -2 & 5 & 6 \\ 6 & -3 & 7 & 8 \\ 8 & -4 & 9 & 10 \end{bmatrix}; b_i = \begin{bmatrix} 5 \\ 7 \\ 9 \\ 11 \end{bmatrix}$
7	$a_{ij} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 1 & 2 \\ -1 & 2 & 4 & 1 \\ 2 & -1 & 1 & -1 \end{bmatrix}; b_i = \begin{bmatrix} 1 \\ 1 \\ 18 \\ -12 \end{bmatrix}$	8	$a_{ij} = \begin{bmatrix} 5 & -3 & 2 & 4 \\ 4 & -2 & 3 & 7 \\ 8 & -6 & -1 & -5 \\ 7 & -3 & 7 & 17 \end{bmatrix}; b_i = \begin{bmatrix} 7 \\ 10 \\ 4 \\ 17 \end{bmatrix}$
9	$a_{ij} = \begin{bmatrix} 1 & 2 & 15 & 11 \\ -2 & 4 & 3 & 61 \\ -3 & -8 & 11 & 12 \\ 15 & 7 & 8 & -4 \end{bmatrix}; b_i = \begin{bmatrix} 2 \\ 4 \\ 5 \\ 9 \end{bmatrix}$	10	$a_{ij} = \begin{bmatrix} 5 & 3 & -4 & -15 \\ 6 & 1 & 10 & 2 \\ 1 & 5 & 3 & 8 \\ -4 & 5 & -1 & 9 \end{bmatrix}; b_i = \begin{bmatrix} 6 \\ 9 \\ 12 \\ -9 \end{bmatrix}$
11	$a_{ij} = \begin{bmatrix} 7 & -8 & 9 & 11 \\ -5 & 6 & 5 & 3 \\ -4 & -7 & 3 & 6 \\ 2 & 1 & 5 & 9 \end{bmatrix}; b_i = \begin{bmatrix} 3 \\ -4 \\ -6 \\ 11 \end{bmatrix}$	12	$a_{ij} = \begin{bmatrix} 1 & 2 & 5 & -11 \\ 2 & -6 & -3 & 8 \\ 3 & 4 & 5 & 1 \\ 7 & 6 & 2 & 20 \end{bmatrix}; b_i = \begin{bmatrix} 7 \\ 11 \\ 15 \\ -9 \end{bmatrix}$
13	$a_{ij} = \begin{bmatrix} 5 & 3 & 3 & -4 \\ -6 & -4 & 2 & 17 \\ 12 & 26 & 1 & 5 \\ 5 & 8 & 9 & 7 \end{bmatrix}; b_i = \begin{bmatrix} 7 \\ -24 \\ 34 \\ -6 \end{bmatrix}$	14	$a_{ij} = \begin{bmatrix} 4 & 7 & 6 & -19 \\ 5 & 9 & 12 & 3 \\ -4 & -3 & 1 & -8 \\ 2 & 5 & 9 & 10 \end{bmatrix}; b_i = \begin{bmatrix} 10 \\ 12 \\ -14 \\ 22 \end{bmatrix}$

15	$a_{ij} = \begin{bmatrix} 12 & 8 & 24 & -5 \\ 16 & 25 & 9 & 9 \\ 2 & 7 & 3 & 4 \\ 21 & -9 & -7 & 6 \end{bmatrix}; b_i = \begin{bmatrix} 5 \\ -17 \\ 22 \\ 3 \end{bmatrix}$	16	$a_{ij} = \begin{bmatrix} -3 & -7 & 16 & 8 \\ -4 & 6 & 12 & 13 \\ 2 & 3 & 7 & 5 \\ 18 & -14 & 1 & 2 \end{bmatrix}; b_i = \begin{bmatrix} 6 \\ -12 \\ 31 \\ 41 \end{bmatrix}$
17	$a_{ij} = \begin{bmatrix} 2 & 5 & 9 & 16 \\ 11 & 16 & 3 & 1 \\ 25 & 14 & -38 & 7 \\ -8 & 9 & -16 & -42 \end{bmatrix}; b_i = \begin{bmatrix} 1 \\ 1 \\ -3 \\ 6 \end{bmatrix}$	18	$a_{ij} = \begin{bmatrix} 2 & -1 & 3 & 2 \\ 3 & 3 & 3 & 2 \\ 3 & -1 & -1 & -2 \\ 3 & -1 & 3 & -1 \end{bmatrix}; b_i = \begin{bmatrix} 4 \\ 6 \\ 6 \\ 6 \end{bmatrix}$
19	$a_{ij} = \begin{bmatrix} 2 & 2 & -1 & 1 \\ 4 & 3 & -1 & 2 \\ 8 & 5 & -3 & 4 \\ 3 & 3 & -2 & 2 \end{bmatrix}; b_i = \begin{bmatrix} 4 \\ 6 \\ 12 \\ 6 \end{bmatrix}$	20	$a_{ij} = \begin{bmatrix} 2 & 3 & 11 & 5 \\ 1 & 1 & 5 & 2 \\ 2 & 1 & 3 & 2 \\ 1 & 1 & 3 & 4 \end{bmatrix}; b_i = \begin{bmatrix} 2 \\ 1 \\ -3 \\ -3 \end{bmatrix}$
21	$a_{ij} = \begin{bmatrix} 2 & 5 & 4 & 1 \\ 1 & 3 & 2 & 1 \\ 2 & 10 & 9 & 9 \\ 3 & 8 & 2 & 2 \end{bmatrix}; b_i = \begin{bmatrix} 20 \\ 11 \\ 40 \\ 37 \end{bmatrix}$	22	$a_{ij} = \begin{bmatrix} 3 & 4 & 1 & 2 \\ 3 & 5 & 3 & 5 \\ 6 & 8 & 1 & 5 \\ 3 & 5 & 3 & 7 \end{bmatrix}; b_i = \begin{bmatrix} -3 \\ -6 \\ -8 \\ -8 \end{bmatrix}$
23	$a_{ij} = \begin{bmatrix} 3.2 & 5.4 & 4.2 & 2.2 \\ 2.1 & 3.2 & 3.1 & 1.1 \\ 1.2 & 0.4 & -0.8 & -0.8 \\ 4.7 & 10.4 & 9.7 & 9.7 \end{bmatrix}; b_i = \begin{bmatrix} 2.6 \\ 4.8 \\ 3.6 \\ -8.4 \end{bmatrix}$	24	$a_{ij} = \begin{bmatrix} 3 & -2 & -5 & 1 \\ 2 & -3 & 1 & 5 \\ 1 & 2 & 0 & -4 \\ 1 & -1 & -4 & 9 \end{bmatrix}; b_i = \begin{bmatrix} 3 \\ -3 \\ -3 \\ 22 \end{bmatrix}$
25	$a_{ij} = \begin{bmatrix} 1 & 1 & -6 & -4 \\ 3 & -1 & -6 & -4 \\ 2 & 3 & 9 & 2 \\ 3 & 2 & 3 & 8 \end{bmatrix}; b_i = \begin{bmatrix} 6 \\ 2 \\ 6 \\ -7 \end{bmatrix}$	26	$a_{ij} = \begin{bmatrix} 2 & -1 & 1 & 2 \\ 6 & -4 & 2 & 4 \\ 6 & -3 & 4 & 8 \\ 4 & -2 & 1 & 1 \end{bmatrix}; b_i = \begin{bmatrix} 2 \\ 3 \\ 9 \\ 1 \end{bmatrix}$
27	$a_{ij} = \begin{bmatrix} 12 & 14 & -15 & 24 \\ 16 & 18 & -22 & 29 \\ 18 & 20 & -21 & 32 \\ 10 & 12 & -16 & 20 \end{bmatrix}; b_i = \begin{bmatrix} 5 \\ 8 \\ 9 \\ 4 \end{bmatrix}$	28	$a_{ij} = \begin{bmatrix} 24 & 14 & 30 & 40 \\ 36 & 25 & 45 & 61 \\ 48 & 28 & 60 & 82 \\ 60 & 35 & 75 & 99 \end{bmatrix}; b_i = \begin{bmatrix} 28 \\ 43 \\ 58 \\ 69 \end{bmatrix}$
29	$a_{ij} = \begin{bmatrix} 5 & -3 & 2 & 4 \\ 4 & -2 & 3 & 7 \\ 8 & -6 & -1 & -5 \\ 7 & -3 & 7 & 17 \end{bmatrix}; b_i = \begin{bmatrix} 3 \\ 1 \\ 9 \\ 15 \end{bmatrix}$	30	$a_{ij} = \begin{bmatrix} 2 & -1 & 3 & 4 \\ 4 & -8 & 5 & 6 \\ 6 & -5 & 7 & 8 \\ 12 & -4 & -9 & 10 \end{bmatrix}; b_i = \begin{bmatrix} 5 \\ 7 \\ 9 \\ 13 \end{bmatrix}$

Аппроксимация функций

Одним из распространенных и практически важных случаев связи между аргументом и функцией является задание этой связи в виде некоторой таблицы $\{x_i; y_i\}$, например, экспериментальные данные. На практике часто приходится использовать табличные данные для приближенного вычисления y при любом значении аргумента x (из некоторой области). Этой цели служит задача о приближении (аппроксимации) функций: данную функцию $f(x)$ требуется приближенно заменить некоторой функцией $g(x)$ так, чтобы отклонение $g(x)$ от $f(x)$ в заданной области было наименьшим. Функция $g(x)$ при этом называется аппроксимирующей. Если приближение строится на заданном дискретном множестве точек $\{x_i\}$, то аппроксимация называется точечной. К ней относятся интерполирование, среднеквадратичное приближение и др. При построении приближения на непрерывном множестве точек (например, на отрезке $[a, b]$) аппроксимация называется непрерывной или интегральной. MATLAB имеет мощные средства точечной и непрерывной аппроксимации с визуализацией результата. Рассмотрим наиболее важную точечную аппроксимацию (обработка экспериментальных данных).

Пример 4. Используя линейную и полиномиальную аппроксимации, получить эмпирические формулы для функции $y=f(x)$, заданной в табличном виде:

x_i	0.75	1.50	2.25	3.00	3.75
y_i	2.50	1.20	1.12	2.25	4.28

Оценить погрешность эмпирических формул.

Протокол программы. В окне команд набираются значения x_i и y_i . Далее выполняется команда построения графика только узловых точек.

```
>> x = [0.75, 1.50, 2.25, 3.00, 3.75] ;
```

```
>> y = [2.50, 1.20, 1.12, 2.25, 4.28] ;
```

```
>> plot (x, y, ' o ');
```

Появляется окно с символами ' o ' на месте узловых точек (рис. 2.1).

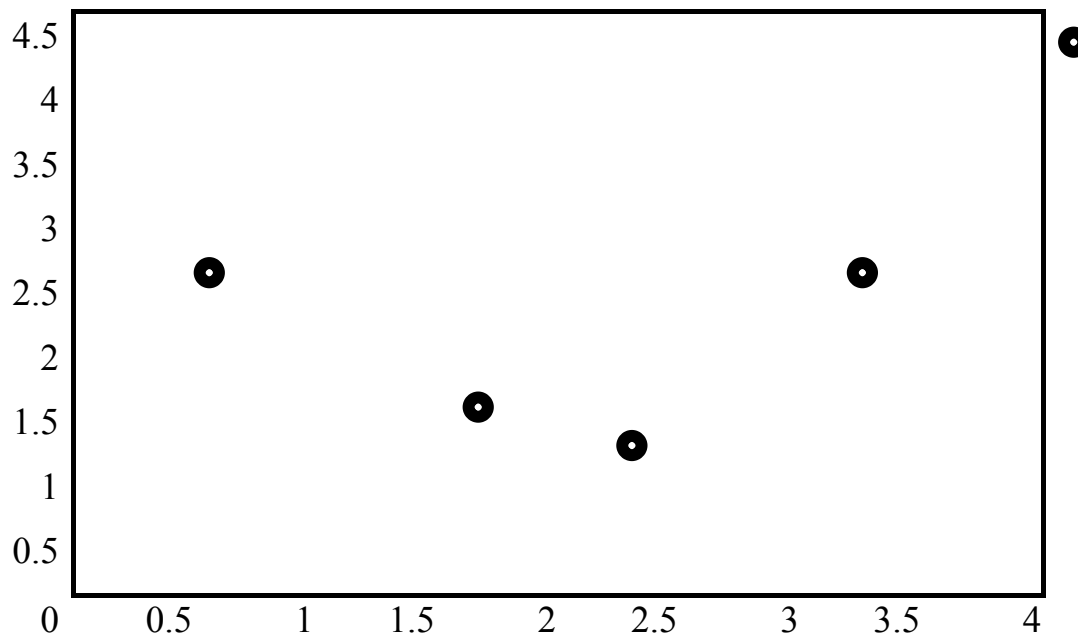


Рис. 2.1

Внимание. Следует помнить, что при полиномиальной аппроксимации максимальная степень полинома на 1 меньше числа экспериментальных точек!

На панели инструментов окна графика узловых точек в меню **Tools** исполняем команду **Basic Fitting**. Появляется окно **Основной Монтаж**. В этом окне птичкой отмечаются необходимые данные аппроксимации. В частности, можно задать следующие операции:

- *показать уравнение* аппроксимирующей функции $y = g(x)$;
- *выбрать метод подбора*: сплайн интерполяции
эрмитовская интерполяция
линейный
квадратные
кубический
.....

В нашей задаче выбираем линейную и полиномиальную аппроксимации. В окне графика появляются соответствующие графики разноцветом и формулы аппроксимирующих функций (рис. 2.2).

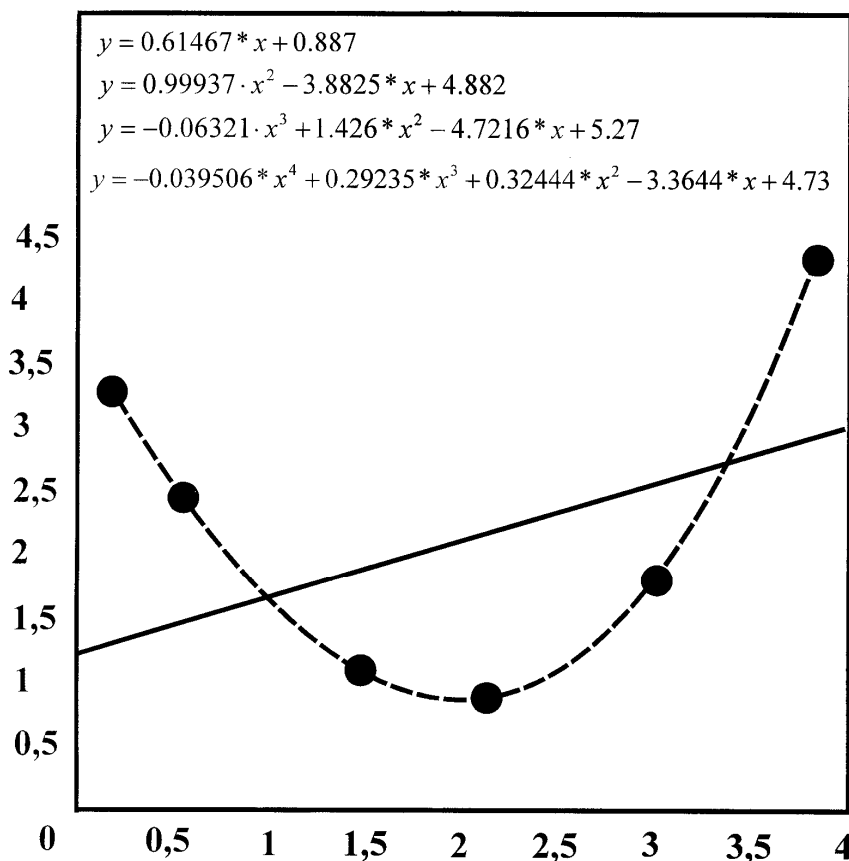


Рис. 2.2

Чтобы узнать погрешность аппроксимации, надо *отметить* птичкой параметр **График остатка** в окне *Основной Монтаж*, и Показать норму остатков. График погрешностей с нормами можно вынести в *отдельное окно*, или вместе с графиком и аппроксимирующих функций – суб-график. Норма погрешностей указывает на статистическую оценку среднеквадратической погрешности. Чем она меньше, тем точнее полученная аппроксимирующая функция $y = g(x)$. В нашем примере:

Linear : norm of residuals (норма погрешности) = 2.1061

Quadratic : norm of residuals = 0.10736

Cubic : norm of residuals = 0.035857

4 th degree : norm of residuals = 9.6305e-015.

График погрешностей можно выводить в виде диаграмм (зоны), линий (линии) или отдельных точек (фрагменты). Сам график погрешностей представляет собой зависимость разности $g(x) - f(x)$ в условных точках, соединенных прямыми линиями.

Кроме линейной и полиномиальной аппроксимации можно выбрать *сплайн- аппроксимацию* – когда на каждом интервале приближения используется *кубический полином* с новыми коэффициентами. В этом случае нельзя получить выражение для аппроксимирующей функции, т. е. такая аппроксимация является неполной. Аналогичными свойствами обладает и *Эрмитовая аппроксимация*. Она имеет только графическую интерпретацию.

Варианты заданий. Получить эмпирические формулы и оценить их погрешность для функции $y = f(x)$, заданной таблично. Данные взять из таблицы 2.3.

Таблица 2.3

1.	x_i	-3	-2	-1	0	1	2	3
	y_i	-0.71	-0.01	0.51	0.82	0.88	0.51	0.49
2.	x_i	-6.6	-5.38	-3.25	-1.76	2.21	3.6	4.5
	y_i	2.89	1.41	0.29	-0.41	-0.69	-0.7	1.2
3.	x_i	0	1	2	3	4	5	6
	y_i	-0.31	0.9	2.11	3.3	4.51	5.73	6.93
4.	x_i	-2	-1	0	1	2	3	4
	y_i	7.1	3.9	1.1	0.8	3.1	4.5	5.3
5.	x_i	-2	-1	-0.5	0	1.5	2	3.5
	y_i	5.9	2.8	2.1	3.2	6.1	7.6	4.3
6.	x_i	-3	-2	-1	0	1	2	3
	y_i	3.1	0.9	0.9	2.8	7.1	6.5	4.1
7.	x_i	0	1	2	3	4	5	6
	y_i	10.0	7.5	5.5	4.0	3.0	2.0	2.24
8.	x_i	-2	-1	0	1.5	2.3	2.6	2.9
	y_i	4.2	5.6	6.8	7.2	9.4	10.5	11.8

9.	x_i	10.0	12.0	13.0	15.0	18.0	20.0	21.0
	y_i	0.66	0.89	1.24	1.36	1.56	1.76	1.92
10.	x_i	22.0	24.0	27.0	30.0	31.0	35.0	40.0
	y_i	1.24	1.37	1.46	1.26	1.66	1.84	1.99
11.	x_i	-7.0	-6.0	-5.0	-4.0	-3.0	-2.0	-1.0
	y_i	22.6	24.7	25.6	24.6	23.5	21.8	19.3
12.	x_i	-25.0	-23.0	-21.0	-18.0	-17.2	-15.4	-14.0
	y_i	0.76	0.74	0.61	0.58	0.84	0.92	1.22
13.	x_i	-4.0	-3.0	-2.0	-1.0	0.0	1.0	2.0
	y_i	1.71	1.56	1.24	1.36	1.78	2.21	4.31
14.	x_i	-22.0	-20.0	-18.0	-16.0	-14.0	-12.0	-10.0
	y_i	-2.26	-1.84	-1.92	-1.76	-1.56	-1.64	-1.34
15.	x_i	23.0	24.0	25.0	26.0	27.0	28.0	29.0
	y_i	1.26	1.37	1.44	1.56	1.15	1.28	1.06
16.	x_i	30.0	33.0	35.0	37.0	39.0	41.0	43.0
	y_i	-2.6	-3.7	-2.5	-4.3	-2.3	-5.6	-1.9
17.	x_i	44.0	45.0	46.0	47.0	48.0	49.0	50.0
	y_i	2.24	3.46	5.36	1.89	1.76	1.54	2.12
18.	x_i	52.0	54.0	56.0	58.0	60.0	62.0	64.0
	y_i	-1.28	-1.33	-1.44	-1.67	-1.77	-2.81	-2.16
19.	x_i	2.2	2.6	3.0	3.4	3.8	4.2	4.6
	y_i	1.88	1.65	1.61	1.73	1.56	1.24	1.99
20.	x_i	5.1	5.3	5.5	5.7	5.9	6.1	6.3
	y_i	-2.8	-3.6	-5.7	-3.4	-1.9	-1.7	-1.5
21.	x_i	7.15	7.35	7.55	7.75	7.95	8.15	8.35
	y_i	-2.2	-3.6	-1.7	-2.8	-1.6	-4.5	-2.2
22.	x_i	9.1	9.2	9.3	9.4	9.5	9.6	9.7
	y_i	1.48	1.16	2.08	1.96	1.81	2.31	5.61
23.	x_i	-10.2	-10.1	-10.0	-9.9	-9.8	-9.7	-9.6
	y_i	-6.5	-7.8	-10.2	-5.4	-4.6	-9.5	-10.3
24.	x_i	11.0	14.0	17.0	20.0	23.0	26.0	29.0
	y_i	1.2	1.6	1.9	1.1	1.16	1.24	1.36
25.	x_i	-50.0	-48.0	-46.0	-44.0	-42.0	-40.0	-38.0
	y_i	1.23	1.32	1.57	1.19	1.16	1.10	2.28
26.	x_i	-36.0	-34.0	-32.0	-30.0	-28.0	-26.0	-24.0
	y_i	1.1	1.3	2.1	1.9	1.7	1.5	1.8
27.	x_i	21.0	23.0	24.0	28.0	31.0	32.0	36.0
	y_i	1.24	1.37	1.56	1.64	1.84	1.26	1.14
28.	x_i	10.0	13.0	17.0	22.0	28.0	35.0	43.0
	y_i	1.21	1.36	1.51	1.84	1.06	1.21	1.36
29.	x_i	-1.0	0.0	3.0	5.0	8.0	12.0	15.0
	y_i	-2.1	-3.6	1.2	-4.3	1.8	2.6	-0.2
30.	x_i	-8.0	-7.0	-5.0	-3.0	-1.0	2.0	5.0
	y_i	1.36	1.88	2.45	-2.1	-10.2	-4.4	1.16

Лабораторная работа №2

Численные методы.

Цель работы:

Изучение наиболее употребимых методов численного решения дифференциальных уравнений и численного интегрирования

Численное решение обыкновенных дифференциальных уравнений.

Многие задачи физики, химии, экологии, механики и других разделов науки и техники при их математическом моделировании сводятся к дифференциальным уравнениям. Поэтому решение дифференциальных уравнений является одной из важнейших математических задач. В вычислительной математике изучаются численные методы решения дифференциальных уравнений, которые особенно эффективны в сочетании с использованием персональных компьютеров.

Среди множества численных методов решения дифференциальных уравнений наиболее простые – это явные одношаговые методы. К ним относятся различные модификации метода Рунге-Кутты.

Постановка задачи:

Требуется найти функцию $y = y(x)$, удовлетворяющую уравнению

$$\frac{dy}{dx} = F(x, y) \quad (2.3)$$

и принимающую при $x = x_0$ заданное значение y_0 :

$$y(x_0) = y_0. \quad (2.4)$$

При этом решение необходимо получить в интервале $x_0 \leq x \leq x_k$. Из теории дифференциальных уравнений известно, что решение $y(x)$ задачи Коши (2.3), (2.4) существует, единственно и является гладкой функцией, если правая часть $F(x, y)$ удовлетворяет некоторым условиям гладкости. Численное решение задачи Коши методом Рунге-Кутты 4-го порядка заключается в следующем. На заданном интервале $[x_0, x_k]$ выбираются узловые точки. Значение решения в нулевой точке известно $y(x_0) = y_0$. В следующей точке $y(x_1)$ определяется по формуле

$$y_1 = y_0 + (k_0 + 2k_1 + 2k_2 + k_3)/6, \quad (2.5)$$

здесь

$$\begin{aligned}
k_0 &= hF(x_0, y_0); & k_1 &= hF(x_0 + h/2, y_0 + k_0/2); \\
k_2 &= hF(x_0 + h/2, y_0 + k_1/2); & k_3 &= hF(x_0 + h, y_0 + k_2);
\end{aligned}
\quad h - \text{шаг сетки},$$

(2.6)

т. е. данный вариант метода Рунге-Кутты требует на каждом шаге четырехкратного вычисления правой части уравнения (2.3). Этот алгоритм реализован в программе **ode45**. Кроме этой программы MATLAB располагает обширным набором аналогичных программ, позволяющих успешно решать обыкновенные дифференциальные уравнения.

Пример 5. Решить задачу Коши

$$\frac{dy}{dx} = 2(x^2 + y^2) \quad \text{на } [0,1] \quad \text{при } y(0) = 1. \quad (2.7)$$

Точное решение имеет вид

$$y(x) = 1.5e^{2x} - x^2 - x - 0.5.$$

Выполним решение данной задачи с помощью программы **ode45**. Вначале в М-файл записываем правую часть уравнения (2.7), сам М-файл оформляется как файл – функция, даем ему имя **F**:

function dydx = F(x, y)

dydx = zeros(1,1);

dydx(1) = 2*(x^2+y(1));

Для численного решения задачи Коши в окне команд набираются следующие операторы.

Протокол программы.

```
>>[X Y] = ode45 ( @ F , [0 1] , [1] ) ;
```

```
% Дескриптор @ обеспечивает связь с файлом – функцией правой части
```

```
% [0 1] – интервал на котором необходимо получить решение
```

```
% [1] – начальное значение решения
```

```
>> plot (X,Y) ;
```

```
>> % Построение графика численного решения задачи Коши (2.7)
```

```
>> hold on; gtext ( ' y(x) ')
```

```
% Команда позволяет с помощью мышки нанести на график надпись y(x)
```

```
>> [X Y]
```

```
>> % Последняя команда выводит таблицу численного решения задачи.
```

Результаты решения. График решения задачи Коши (2.7) показан на рис. 2.3. Численное решение представлено в таблице 2.4, где приведены только отдельные узловые точки. В программе **ode45** по умолчанию интервал разбивается на 40 точек с шагом $h = 1/40 = 0.025$.

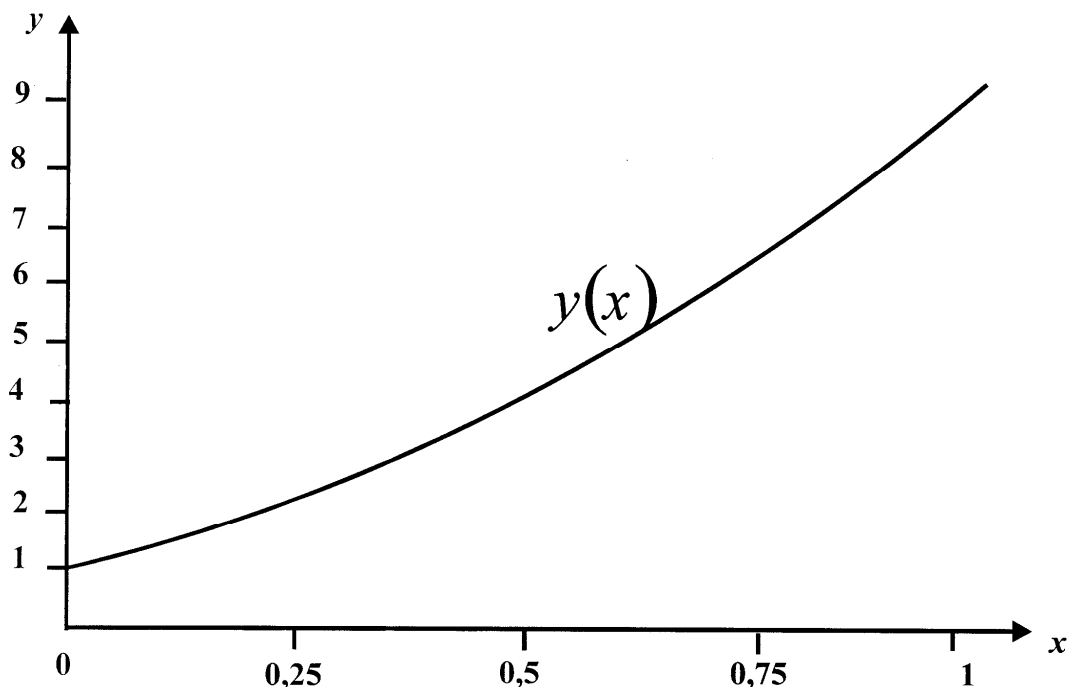


Рис. 2.3

Таблица 2.4

x_i	Метод Рунге-Кутта	Точное решение
0.0	1.0	1.0
0.1	1.2221	1.2221
0.2	1.4977	1.4977
0.3	1.8432	1.8432
0.4	2.2783	2.2783
0.5	2.8274	2.8274
0.6	3.5202	3.5202
0.7	4.3928	4.3928
0.8	5.4895	5.4895
0.9	6.8645	6.8645
1.0	8.5836	8.5836

Как следует из таблицы 2.4 численное решение программой **ode45** является точным.

Варианты заданий. Построить график и вывести в виде таблицы решение задачи Коши на интервале $[0; 1]$ методом Рунге-Кутта 4-го порядка. Данные взять из таблицы 2.5.

Таблица 2.5

№ п/п	$f(x,y)$	y_0
1.	$x^3 \sin y + 1$	0.0
2.	$x^2 \sin y - 1$	0.1
3.	$e^{+x} + 3y$	2.0
4.	$\sqrt{y^2 + x^3}$	0.3
5.	$\sqrt{y^3 + x^2}$	0.4
6.	$1/(1 + y^2) + x^2$	0.0
7.	$1/(1 + y^2) + xy$	0.1
8.	$\cos y + xy$	0.2
9.	$x^2 \cos y + 0.1$	0.3
10.	$x^3 \cos y + 0.1$	0.4
11.	$\cos(xy) - 0.5$	0.5
12.	$e^{-y} + e^x - 2$	0.0
13.	$e^{-y} - e^x - 0.1$	0.5
14.	$e^{-xy} + 1$	0.4
15.	$\sqrt{y^2 + x^4}$	0.3
16.	$(xy)^2 + \sqrt{x}$	0.2
17.	$\cos(x + y) + x^2$	0.1
18.	$\sin(x + y) - x^2$	0.0
19.	$\sin(xy) + 1$	0.1
20.	$\sin y + xy$	0.2
21.	$(x^2 + y^2)x$	0.3
22.	$e^x(1 + xy)$	0.4
23.	$e^y(1 + xy)$	0.5
24.	$\ln(1 + x^2) + y$	0.6
25.	$1 + ye^{x^2}$	0.7
26.	$\sin x^2 + y^2$	0.0
27.	$\cos x^2 + xy$	0.1
28.	$\sin[y/(1 + x^2)] + 1$	0.2
29.	$\ln(1 + y^2) + 1$	0.3
30.	$1 + xe^{y^2}$	0.4

Приближенное вычисление определенных интегралов

К вычислениям определенных интегралов сводятся многие практические задачи физики, химии, экологии, механики и других естественных наук. На практике формулой Ньютона-Лейбница не всегда удастся воспользоваться. В этом случае используются методы численного интегрирования. Они основаны на следующих соображениях: с геометрической точки зрения определенный интеграл

$\int_a^b f(x)dx$ представляет собой площадь криволинейной трапеции. Идея

методов численного интегрирования сводится к разбиению интервала $[a; b]$ на множество меньших интервалов и нахождению искомой площади как совокупности элементарных площадей, полученных на каждом частичном промежутке разбиения. В зависимости от использованной аппроксимации получаются различные формулы численного интегрирования, имеющие различную точность.

Рассмотрим методы трапеций и Симпсона (парабол).

Метод трапеций.

Здесь используется линейная аппроксимация, т. е. график функции $y = f(x)$ представляется в виде ломаной, соединяющей точки y_i . Формула трапеций при постоянном шаге $h = \frac{b-a}{n}$, где n – число участков, имеет

вид

$$\int_a^b f(x)dx = h \left(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i \right). \quad (2.8)$$

В MATLAB данную формулу реализует программа ***trapz(x,y)***.

Метод Симпсона

Если подынтегральную функцию заменить параболой, то формула Симпсона с постоянным шагом интегрирования предстанет в виде

$$\int_a^b f(x)dx = \frac{h}{3} [y_0 + 4(y_1 + y_3 + \dots + y_{n-1}) + 2(y_2 + y_4 + \dots + y_{n-2}) + y_n]. \quad (2.9)$$

В MATLAB формула Симпсона реализуется программой ***quad***.

Подынтегральная функция может задаваться с помощью дескриптора @, тогда она программируется в файле – функции, или с помощью апострофов, тогда она записывается в самой программе ***quad***. Точность вычисления интегралов по умолчанию принята равной $1 \cdot 10^{-6}$.

Пример 6. Вычислить и вывести на печать по методам трапеций и Симпсона значения интеграла

$$\int_0^1 \frac{dx}{1+x^2}$$

Протокол программы метода трапеций

```
>> x = 0 : 0.0001 : 1.0 ;
```

```
>> y = 1./ (1+x.^2) ;
```

```
>> z = trapz(x, y)
```

Результат вычислений

z =

0.7854

Протокол программы метода Симпсона

```
>> quad ( ' (1./(1+x.^2)) ', 0, 1)
```

Результат вычислений

ans =

0.7854

Точное значение интеграла равно 0.785398163.

Как видно из примера 6 полученные результаты являются практически точными, а сами протоколы весьма просты.

Варианты заданий. Вычислить и вывести на печать значения определенного интеграла методами трапеций и Симпсона. Данные взять из таблицы 2.6.

Таблица 2.6

№ п/п	Подынтегральная функция f(x)	Интервал интегрирования [a; b]	Точность вычислений интеграла
1	$\ln x / x \sqrt{1 + \ln x}$	[1; 3.5]	0.001
2	$tg^2 x + ctg^2 x$	$[\pi/6; [\pi/3]$	0.002
3	$1/x \ln x$	[1.5; 3.]	0.0001
4	$\ln^2 x / x$	[1.0; 4,0]	0.003
5	$\sqrt{e^x - 1}$	[0; ln2]	0.0015
6	$xe^x \sin x$	[1.0; 4.0]	0.002
7	$x \frac{e^x - e^{-x}}{2}$	[0.0; 2.0]	0.001
8	$1/\sqrt{9+x^3}$	[2.0; 5.0]	0.001
9	$\sin(1/x)x^4$	[1.0; 2.5]	0.0005
10	$x^3 \arctg x$	$[0.0; \sqrt{3}]$	0.003
11	$\arcsin \sqrt{x/(1+x)}$	[0.0; 3,0]	0.001
12	$x^2(1 + \ln x)$	[1.5; 3.0]	0.0025
13	$1/\sqrt{1+3x+2x^2}$	[0,0; 5.0]	0.001
14	$\sqrt{x^2 - 0.14} / x$	[2.3; 6.0]	0.002

15	$2^{3x} \ln \cos x $	$[0.0; \pi/2]$	0.001
16	$(e^{3x} + 1)/(e^x + 1)$	$[0.0; 2.0]$	0.0015
17	$x \arctg x / \sqrt{1+x^2}$	$[0.0; 2.0]$	0.002
18	$\sin^5 x / \ln(1+x^3)$	$[0.0; \pi/4]$	0.001
19	$x^2 \sqrt{4-x^2}$	$[0.0; 1.8]$	0.0006
20	$e^x \cos^2 x$	$[0.0; \pi]$	0.0016
21	$(1.5x^2 + x)/(x^3 + 2)$	$[0.0; 1.2]$	0.002
22	$(\sin - x^3)/(x^2 + 2.7)$	$[2.0; 4.4]$	0.0014
23	$(3.5 \lg x + x)/(x^3 + 3.7)$	$[0.0; 1.2]$	0.002
24	$(1.2 - 3.2x^2)/(1 + \sin^2 x)$	$[2.0; 4.4]$	0.0011
25	$(3.17x + 4.2)/\cos^2 x$	$[1.0; 2.2]$	0.0023
26	$(x^2 + 3.2)/(x^5 + 4.7)$	$[0.0; 1.8]$	0.0015
27	$(1.5x^2 + x)/(x^3 + 2)$	$[0.0; 1.2]$	0.001
28	$e^x \ln \sin x^2 $	$[1.0; 3.0]$	0.002
29	$\lg x^2 / e^{-x^2}$	$[0.0; 1.0]$	0.0013
30	$(1.6x - 2.7)/(1.5x^3 + 3.9)$	$[1.0; 2.2]$	0.0025

Численное решение нелинейных уравнений

Задача нахождения корней нелинейных уравнений встречается в различных областях научно-технических исследований. Проблема формулируется следующим образом. Пусть задана непрерывная функция $f(x)$ и требуется найти корень уравнения

$$f(x) = 0.$$

Будем предполагать, что имеется интервал изменения x $[a; b]$, на котором необходимо исследовать функцию $f(x)$ и найти значение x_0 , при котором $f(x_0)$ равно или весьма мало отличается от нуля.

Данная задача в системе MATLAB может быть решена следующим образом. Вначале необходимо построить график функции $f(x)$ на заданном интервале и убедиться в существовании корня или нескольких корней. Затем применить программы поиска корней. Если существует *один* корень и график $f(x)$ *пересекает* ось ox , то можно применить программу **fzero**. Если $f(x)$ имеет больше одного корня и может касаться и пересекать ось ox , то следует применить более мощную программу **fsolve** из пакета **Optimization Toolbox**, которая решает задачу методом наименьших квадратов. Программа **fzero** использует известные численные методы: деление отрезка пополам, секущей и обратной квадратичной интерполяции.

Пример 7. Найти корень нелинейного уравнения $10^x + 2x - 100 = 0$ на интервале $[1.0; 2.0]$.

Протокол программы

```
>> % Строим график заданной функции
```

```
>> x = 1.0 : 0.001 : 2.0; y = 10.0.^x + 2.0*x - 100.0;
```

```
>> plot (x, y) ; grid on
```

Появляется окно с графиком функции $10^x + 2x - 100$ (см. рис. 2.4), из которого следует, что корень функции на заданном интервале существует. Для точного определения корня применяем *fzero* и *fsolve*.

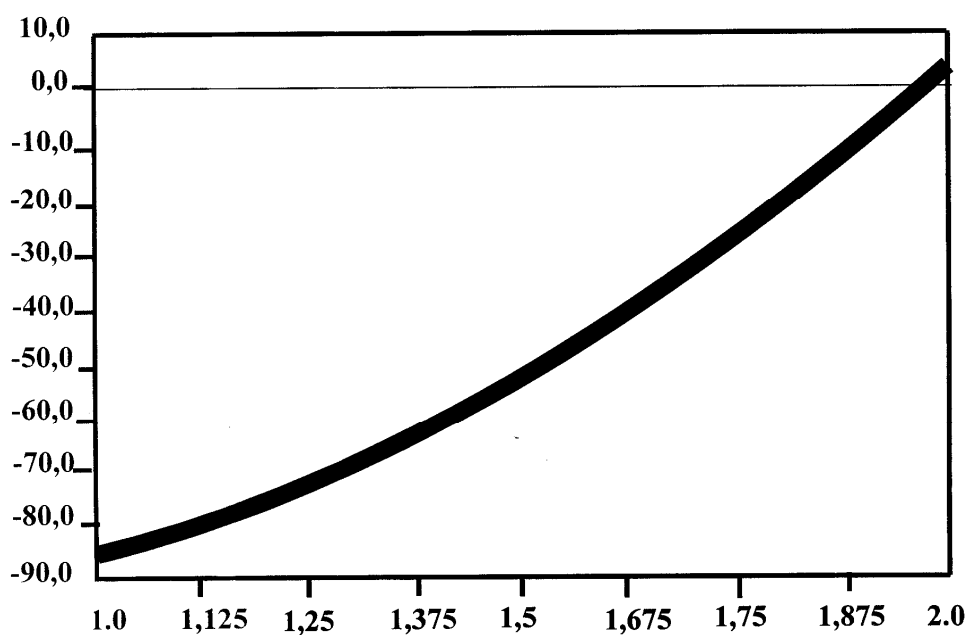


Рис. 2.4

```
>> X1 = fzero ( ' (10.0.^x + 2.0*x - 100.0) ', [1.0 2.0])
```

Результат решения

X1 =

1.9824

```
>> X2 = fsolve ( ' (10.0.^x + 2.0*x - 100.0) ', 1.0 : 2.0)
```

Результат решения

X2 =

1.9824 1.9824

Варианты заданий. Построить график и найти корень нелинейного уравнения. Данные взять из таблицы 2.7.

Таблица 2.7

№ п/п	Уравнение $f(x) = 0$	Отрезок [a; b]
1	$x \arctg - 1 = 0$	[1.0; $\sqrt{3}$]
2	▪ EMBED Equation.3	[2.0; 3.0]
3	$x^3 - 9x^2 + 5x - 6 = 0$	[8.0; 9.0]
4	$e^x - \frac{1}{x} - 1 = 0$	[0.5; 1.0]
5	$\arctg 2x - \frac{1}{1+x} = 0$	[0.0; 1.0]
6	$e^x - \ln x - 20 = 0$	[3.0; 3.2]
7	$\sqrt{x} - \tg x(1-x) = 0$	[0.0; 1.0]
8	$\sin \sqrt{x} - \cos \sqrt{x} + 2\sqrt{x} = 0$	[0.0; 0.2]
9	$x^4 + 2x^3 - x - 1 = 0$	[0.8; 1.0]
10	$x^3 - e^{4x} - 5.5 = 0$	[2.6; 3.0]
11	$x^6 - 3x^2 + x - 1 = 0$	[1.0; 1.5]
12	$\sqrt[3]{5-x} - x = 0$	[1.0; 2.0]
13	$x^2 - \ln x = 0$	[0.0; 1.0]
14	$x^2 - \cos x = 0$	[0.0; 1.0]
15	$\ln x - \arctg x = 0$	[3.0; 4.0]
16	$x^2 \arctg x - 1 = 0$	[1.0; 1.2]
17	$x^2 + \ln x - 4 = 0$	[1.0; 2.0]
18	$x - \arctg \sqrt[3]{x} = 0$	[0.0; 1.0]
19	$x^2 - e^x - 2 = 0$	[-0.2; -0.1]
20	$x^2 - \ln(x+1) = 0$	[0.1; 0.9]
21	$x^5 - x - 2 = 0$	[1.0; 1.4]
22	$x - 2 - \sqrt[4]{x} = 0$	[3.0; 4.0]
23	$x - \tg x = 0$	[0.0; 1.5]
24	$x + \ln x - 0.5 = 0$	[0.0; 1.0]
25	$\ln x + \sqrt{x} = 0$	[0.1; 1.0]
26	$\sqrt{x} - \cos \sqrt{x} = 0$	[0.4; 0.6]
27	$x^2 - \ln(1+x^2) - 9.75 = 0$	[3.0; 4.0]
28	$x + \sqrt[3]{x} - 6.09 = 0$	[4.0; 5.0]
29	$x^3 - \sqrt{x} - 9.5 = 0$	[2.0; 3.0]
30	$\arccos 2x - x^2 - 0.35 = 0$	[0.0; 0.48]

Численное решение оптимизационных задач

Под оптимизацией понимают процесс выбора наилучшего варианта из всех возможных. С точки зрения инженерных расчетов методы оптимизации позволяют выбрать наилучший вариант конструкции,

наилучшее распределение ресурсов, минимальный урон природной среде и т. п. В процессе решения задачи оптимизации необходимо найти оптимальные значения некоторых параметров, их называют проектными параметрами. Выбор оптимального решения проводится с помощью некоторой функции, называемой целевой функцией. Целевую функцию можно записать в виде

$$u = f(x_1, x_2, \dots, x_n), \quad (2.10)$$

где x_1, x_2, \dots, x_n – проектные параметры.

Можно выделить 2 типа задач оптимизации – *безусловные* и *условные*. Безусловная задача оптимизации состоит в отыскании максимума или минимума функции (2.10) от n действительных переменных и определении соответствующих значений аргументов на некотором множестве G n -мерного пространства. Обычно рассматриваются задачи минимизации; к ним легко сводятся и задачи на поиск максимума путем замены знака целевой функции на противоположный. Условные задачи оптимизации – это такие, при формулировке которых задаются некоторые условия (ограничения) на множестве G . Здесь рассмотрим только безусловные задачи оптимизации.

Поиск минимума функции одной переменной.

Для решения этой задачи используются методы золотого сечения или параболической интерполяции (в зависимости от формы задания функции), реализованные в программе *fminbnd*.

Пример 8. Найти и вывести на печать минимальное значение функции $f(x) = 24 - 2x/3 + x^2/30$ на $[5; 20]$.

Строим график этой функции, чтобы убедиться в наличии минимума на заданном интервале.

Протокол программы

```
>> x = 5.0 : 0.001 : 20.0 ; y = 24 - 2* x/3 + x.^2/30 ;
```

```
>> plot(x, y) ; grid on
```

Появляется окно с графиком этой функции (рис. 2.5), где отмечаем наличие минимума.

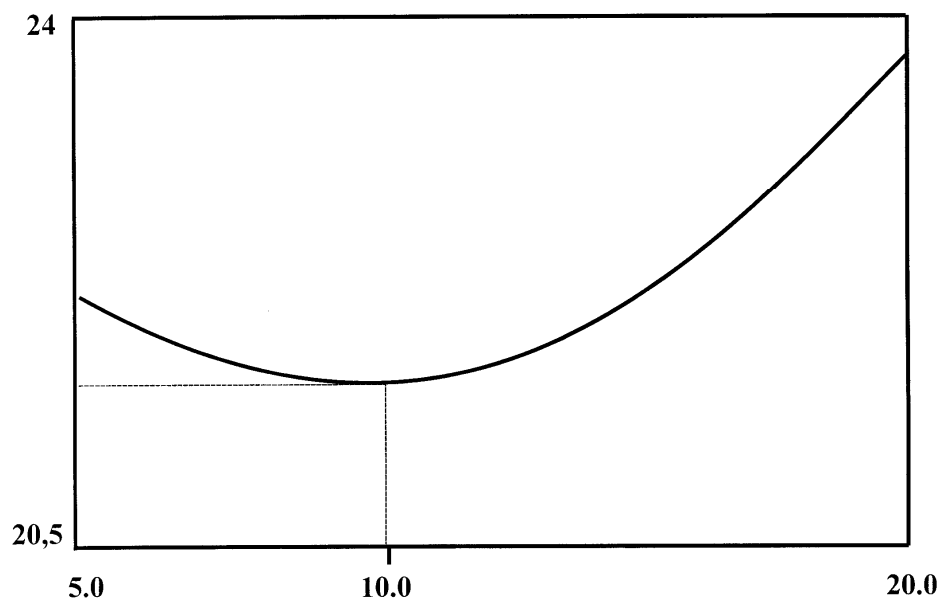


Рис. 2.5

Далее, для точного определения координаты и значения минимума привлекаем программу *fminbnd*.

```
>> [x, y] = fminbnd ( ' (24.0 - 2* x/3 + x.^2/30) ', 5.0, 20.0)
```

Результат поиска

$x =$

10.0000

$y =$

20.6667

Варианты заданий. Найти и вывести на печать координату и минимальное значение функции $f(x)$ на $[a; b]$. Данные взять из таблицы 2.8.

Таблица 2.8

№ п/п	Функция $f(x)$	Отрезок $[a; b]$
1	$f(x) = x / \ln x$	$[1.2; 4]$
2	$f(x) = x - 2 \sin x$	$[0; \pi/2]$
3	$f(x) = (x^2 - 1)/(x^2 + 1)$	$[-2; 2]$
4	$f(x) = 3x^4 - 4x^3 + 12x^2 + 1$	$[-2; 2]$
5	$f(x) = x - 2 \ln x$	$[1; 3]$
6	$f(x) = e^x \cos x$	$[\pi; 3\pi/2]$
7	$f(x) = (1 - x + x^2)/(1 + x - x^2)$	$[0; 1]$
8	$f(x) = -\sqrt{2x - x^2}$	$[0; 2]$
9	$f(x) = (x - 2)^5 (2x + 1)^4$	$[-0.5; 1.5]$
10	$f(x) = x^x$	$[0,1; 1.0]$

11	$f(x) = e^{-1/x^2}$	$[-0.5; 0.5]$
12	$f(x) = x\sqrt{1-x^2}$	$[-1.0; 0]$
13	$f(x) = ((e^x + e^{-x})/2)^3 + 1$	$[-0.5; 0.5]$
14	$f(x) = -x/(x^3 + 2)$	$[0.5; 1.5]$
15	$f(x) = x^2 / \sqrt[3]{x^3 - 4}$	$[1.6; 2.2]$
16	$f(x) = \sqrt{ x^2 - 3 } / x$	$[1; 2]$
17	$f(x) = x^2 \sqrt{ x^2 - 1 }$	$[1.1; 1.6]$
18	$f(x) = 1/(\sin x + \cos x)$	$[0; \pi/3]$
19	$f(x) = x/2 + \arctg x$	$[0.5; 1.2]$
20	$f(x) = xe^{-x^2/2}$	$[-1.5; -0.5]$
21	$f(x) = e^{-1x^2} / x$	$[-2.0; -1.0]$
22	$f(x) = - x ^3 e^{-x^2/2}$	$[-2.0; -1.0]$
23	$f(x) = x^2 \ln x$	$[0.1; 1.0]$
24	$f(x) = x \ln^2 x $	$[-0.05; -0.2]$
25	$f(x) = x \arctg x$	$[-0.5; 0.5]$
26	$f(x) = \sin x + \cos x$	$[\pi; 3\pi/2]$
27	$f(x) = -\ln x / x^2$	$[1.0; 2.0]$
28	$f(x) = x^2 \ln^2 x$	$[0.1; 0.5]$
29	$f(x) = \sin x / x$	$[\pi; 2\pi]$
30	$f(x) = -x^{1/x}$	$[2.0; 3.0]$

Поиск минимума функций нескольких переменных

Данная задача значительно сложнее первой. Рассмотрим ее решение на примере функции двух переменных. Алгоритм может быть распространен на функции большего числа переменных. Для минимизации функций нескольких переменных MATLAB использует симплекс – метод Нелдера-Мида. Данный метод является одним из лучших методов поиска минимума функций многих переменных, где не вычисляются производные или градиент функции. Он сводится к построению симплекса в n -мерном пространстве, заданного $n + 1$ вершиной. В двухмерном пространстве симплекс является треугольником, а в трехмерном – пирамидой. На каждом шаге итераций выбирается новая точка решения внутри или вблизи симплекса. Она сравнивается с одной из вершин симплекса. Ближайшая к этой точке вершина симплекса заменяется этой точкой. Таким образом, симплекс

перестраивается и позволяет найти новое, более точное положение точки решения. Алгоритм поиска повторяется, пока размеры симплекса по всем переменным не станет меньше заданной погрешности решения. Программу, реализующую симплекс-методы Нелдера-Мида, удобно использовать в следующей записи

[x, min f] = fminsearch (...),

где x – вектор координат локального минимума;

$\min f$ – значение целевой функции в точке минимума.

Саму целевую функцию удобно представить с помощью дескриптора @ в М-файле.

Пример 9. Найти и вывести на печать координаты и значение минимума функции двух переменных $f(x, y) = (x^2 + y^2 - 3)^2 + (x^2 + y^2 - 2x - 3)^2 + 1$, если начальная точка поиска имеет координаты $M_0(1; 1)$.

Анализ функции показывает, что $\min f = 1$ $x = 0$, $y = \sqrt{3} = 1.73205$.

Строим трехмерный график этой функции, чтобы убедиться в наличии минимума. Возьмем интервал $x \in [-1; 1]$; $y \in [1; 3]$.

Протокол программы

```
>> [X,Y] = meshgrid ( [-1 : 1, 1 : 3] );
```

```
>> Z = (X.^2 + Y.^2 - 3).^2 + (X.^2 + Y.^2 - 2*X - 3).^2 + 1 ;
```

```
>> plot 3 (X,Y,Z)
```

После построения трехмерного графика выполняем поиск минимума. В М-файле программируем целевую функцию

```
function f = Fxy (x)
```

```
f = (x(1) ^ 2 + x(2) ^ 2 - 3)^2 + (x(1) ^ 2 + x(2) ^ 2 - 2*x(1) - 3)^2 + 1;
```

Решаем поставленную задачу в окне команд

```
>> [xmin, minf] = fminsearch ( @ Fxy, [1; 1] )
```

Результаты поиска

```
xmin =  
- 0.0000    1.7320
```

```
minf =  
1.0000
```

Как видно, результаты решения задачи точные.

Варианты заданий. Найти и вывести на печать координаты и минимальное значение функции двух переменных. Поиск начать с точки $M_0(x_0, y_0)$. Данные взять из таблицы 2.9.

Таблица 2.9

№ п/п	Функция $f(x, y)$	Координаты начальной точки M_0 (x_0, y_0) .
1	2	3
1	$(2x^2 - y - 3)^2 + x^2 + 2x + 2$	(1; 1)
2	$(xy + 2)^2 + y^2 + 2y + 4$	(2; 2)
3	$(x^2 y^2 - y + 2)^2 + x^2 + 1$	(2; 2)
4	$(3x^2 + 2y^2 - 1)^2 + (xy - 3)^2$	(2; 2)
5	$(2x^2 - 7y^2 - 2)^2 + (x^2 + y^2 - 20)^2 + 3$	(2; 2)
6	$(x^2 + y^2 - 2x - 3)^2 + (x^2 + y^2 - 2y - 3)^2$	(2; 2)
7	$(x^2 - 6x + y^2 + 8)^2 + x^2 y^2 + 1$	(2; 2)
8	$(x^2 - y - 2)^2 + (x - y + 3)^2$	(2; 2)
9	$\ln(1 + x^2 + y^2)^2 + (x - y - 1)^2$	(2; 2)
1	2	3
10	$(x^2 + y^2 - 1)^2 + (x^2 - 6x + y^2 + 8)^2$	(2; 2)
11	$x^3 + y^3 - 3xy$	(0.5; 0.5)
12	$x^2 + xy + y^2 - 3x - 6y$	(0.5; 3.5)
13	$-xy^2(1 - x - y)$	(0; 0)
14	$3x^2 - x^3 + 3y^2 + 4y$	(0.1; -1.0)
15	$xy + 50/x + 20/y$	(4; 1)
16	$x^2 + y^2 - 2\ln x - 18\ln y$	(0.5; 2.5)
17	$x^3 + 3xy^2 - 15x - 12y$	(1.5; 0.5)
18	$2x^3 - xy^2 + 5x^2 + y^2$	(0.5; 0.5)
19	$(2x^2 + y^2)e^{-(x^2+y^2)}$	(0.3; 0.3)
20	$-2 + \sqrt[3]{x^2 + y^2}$	(0.25; 0.25)
21	$(x - 1)^2 + (y - 2)^2 - 1$	(0.5; 1.5)
22	$\sin(x^2 + y^2 - 0.5)$	(0.5; 0.5)
23	$x^2 - xy + y^2 + 2x - 2y + 1$	(-1.5; 0.5)
24	$xy(x + y - 4)$	(1.0; 1.0)
25	$x^3 y^2 (x + y - 5)$	(2.0; 1.5)
26	$x^2 + xy + y^2 + 1/x + 1/y$	(0.2; 0.3)
27	$-(\sin x + \sin y + \sin(x + y))$	($\pi/4$; $\pi/4$)
28	$-\sin x \sin y \sin(x + y)$	($\pi/4$; $\pi/4$)

№ п/п	Функция $f(x, y)$	Координаты начальной точки M_0 (x_0, y_0).
29	$x^3 y^3 - 9xy + 1$	(2.5; 2.5)
30	$x^4 + y^4 - 2x^2 + 4xy - 2y^2 + 1$	(1.0; -1.0)

Литература

1. Дьяконов, Владимир Павлович. MATLAB 6/6.1/6.5 + Simulink 4/5 : Основы применения : Полное руководство пользователя / В. П. Дьяконов. — М. : СОЛОН-Пресс, 2002. — 768 с
2. Лазарев, Юрий Ф. MatLAB 5.x / Ю. Ф. Лазарев. — Киев : BHV : Ирина, 2000. — 383 с.
3. Поршнев, Сергей Владимирович. MATLAB 7: основы работы и программирования : учебное пособие для вузов / С. В. Поршнев. — М. : Бином, 2006. — 320 с.
4. Кетков, Юлий Лазаревич. MATLAB 7. Программирование, численные методы / Ю. Л. Кетков, А. Ю. Кетков, М. М. Шульц. — СПб. : БХВ-Петербург, 2005. — 752 с.

Оглавление

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ. ЭЛЕМЕНТЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ И ВИЗУАЛИЗАЦИИ РАСЧЕТОВ В СИСТЕМЕ MATLAB.	3
АЛФАВИТ ЯЗЫКА ПРОГРАММИРОВАНИЯ	3
АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОПЕРАТОРЫ	3
ЭЛЕМЕНТАРНЫЕ ФУНКЦИИ	5
ПОНЯТИЕ О ФАЙЛАХ-СЦЕНАРИЯХ И ФАЙЛАХ-ФУНКЦИЯХ.....	6
ОСНОВЫ ПРОГРАММИРОВАНИЯ	8
<i>Оператор присваивания.....</i>	<i>8</i>
<i>Перенос строки.....</i>	<i>8</i>
<i>Ввод и вывод данных.....</i>	<i>9</i>
<i>Форматы чисел.....</i>	<i>9</i>
<i>Формирование векторов и матриц.....</i>	<i>10</i>
<i>Типы данных</i>	<i>11</i>
<i>Оператор двоеточие :</i>	<i>12</i>
<i>Оператор разветвления if.....</i>	<i>12</i>
<i>Операторы циклов.....</i>	<i>13</i>
<i>Сообщения об ошибках и исправление ошибок.....</i>	<i>14</i>
<i>Вычисление определителя квадратной матрицы.....</i>	<i>14</i>
<i>Обращение матриц.....</i>	<i>14</i>
ОСНОВЫ ГРАФИЧЕСКОЙ ВИЗУАЛИЗАЦИИ ВЫЧИСЛЕНИЙ.....	15
<i>Построение графиков отрезками прямых.....</i>	<i>15</i>
<i>Создание массивов данных для трехмерной графики</i>	<i>16</i>
<i>Построение графиков поверхностей</i>	<i>17</i>
<i>Включение и выключение масштабной сетки</i>	<i>17</i>
<i>Представление нескольких графиков в одном окне</i>	<i>18</i>
<i>Ввод текста на график с помощью мыши</i>	<i>18</i>
<i>Управление свойствами осей графиков.....</i>	<i>18</i>
ЛАБОРАТОРНАЯ РАБОТА №1.....	19
МАТЛАВ В ЗАДАЧАХ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ	19
ТАБУЛИРОВАНИЕ ФУНКЦИЙ.....	19
РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ МЕТОДОМ ИСКЛЮЧЕНИЯ ГАУССА	23
АППРОКСИМАЦИЯ ФУНКЦИЙ	28
ЛАБОРАТОРНАЯ РАБОТА №2.....	33
ЧИСЛЕННЫЕ МЕТОДЫ.	33

ЧИСЛЕННОЕ РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ.	33
ПРИБЛИЖЕННОЕ ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ	37
ЧИСЛЕННОЕ РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ	39
ЧИСЛЕННОЕ РЕШЕНИЕ ОПТИМИЗАЦИОННЫХ ЗАДАЧ.....	41
ПОИСК МИНИМУМА ФУНКЦИЙ НЕСКОЛЬКИХ ПЕРЕМЕННЫХ	44
ЛИТЕРАТУРА.....	48

Учебное издание

ШЕСТАКОВ Василий Васильевич

ВВЕДЕНИЕ В «MATLAB»

Методические указания к выполнению лабораторных работ
по курсу «Компьютерные технологии в приборостроении»
для студентов II курса, обучающихся по направлению
200100 «Приборостроение»


**Отпечатано в Издательстве ТПУ в полном соответствии
с качеством предоставленного оригинал-макета**

Подписано к печати _____.2010. Формат 60х84/16. Бумага «Снегурочка».
Печать XEROX. Усл.печ.л. 9,01. Уч.-изд.л. 8,16.
Заказ _____. Тираж _____. экз.



Национальный исследовательский Томский политехнический
университет
Система менеджмента качества
Томского политехнического университета сертифицирована
NATIONAL QUALITY ASSURANCE по стандарту ISO 9001:2008



ИЗДАТЕЛЬСТВО  ТПУ. 634050, г. Томск, пр. Ленина, 30
Тел./факс: 8(3822)56-35-35, www.tpu.ru