Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра «Вычислительные методы и программирование»

А. З. Самуйлов, Т. М. Кривоносова

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА С MATLAB

Методическое пособие
по дисциплине «Вычислительная математика»
для студентов специальностей
1-39 02 01 «Моделирование и компьютерное проектирование радиоэлектронных средств», 1-39 01 01 «Радиотехника»,
1-39 01 03 «Радиоинформатика»,
1-38 02 03 «Техническое обеспечение безопасности»
всех форм обучения

Минск БГУИР 2011

УДК 519.6:004.42(076)
ББК 22.19я73+32.973.26-018.2я73
С17

Р е ц е н з е н т:
заведующий кафедрой «Защита информации» учреждения образования
«Белорусский государственный университет информатики
и радиоэлектроники», доктор технических наук,
профессор Л. М. Лыньков

Самуйлов, А. З.

С17 Вычислительная математика с *MATLAB* : метод. пособие по дисц. «Вычислительная математика» для студ. спец. 1-39 02 01 «Моделирование и компьютерное проектирование радиоэлектронных средств», 1-39 01 01 «Радиотехника», 1-39 01 03 «Радиоинформатика», 1-38 02 03 «Техническое обеспечение безопасности» всех форм обуч. / А. З. Самуйлов, Т. М. Кривоносова. – Минск : БГУИР, 2011. – 48 с. : ил.

ISBN 978-985-488-672-5.

Изложены основы работы в системе компьютерной математики *MATLAB* с примерами численных и аналитических вычислений, графика *MATLAB*, программирование средствами *MATLAB*.

Даны краткие теоретические сведения об алгоритмах вычислительной математики и их реализации в *MATLAB* с примерами и задачами для самостоятельной работы.

Рассмотрены алгоритмы решения уравнений с одной неизвестной, решения систем линейных и нелинейных алгебраических уравнений, систем обыкновенных дифференциальных уравнений, алгоритмы аппроксимации функций.

УДК 519.6:004.42(076)
ББК 22.19я73+32.973.26-018.2я73

ISBN 978-985-488-672-5

СОДЕРЖАНИЕ

1. ВВОД И ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ АЛГЕБРАИЧЕСКИХ ВЫРАЖЕНИЙ

Аббревиатура *MATLAB* получена из слов *MATrix LABoratory* – матричная лаборатория; это означает, что система компьютерной математики *MATLAB* создана для работы с матрицами, и даже обычные числа и переменные используются системой как матрицы размером 1×1. Однако ориентация на матричные вычисления почти не ощущается, если матриц в выражениях нет.

1.1. Вычисления в командном режиме (режиме калькулятора)

После загрузки система *MATLAB* готова к работе.

Вычисления можно проводить в командном режиме (по умолчанию) или создать программу вычислений. Для создания программы используется специальный редактор файлов.

Рассмотрим командный режим работы.

Строки ввода команд помечаются знаком **>>**. Типичная команда присваивает переменной, значение которой необходимо вычислить, результат вычисления выражения.

Выражение составляется из операторов, функций и имен переменных с помощью знаков математических операций +, –, *, /, ^ (^ – операция возведения в степень). Вычисление выражения производится после нажатия клавиши *Enter* (*Ввод*). Процесс вычисления можно прервать нажатием комбинации клавиш *Ctrl + Break.*

На одной строке может быть несколько выражений, разделенных запятой или точкой с запятой. Если имя переменной и знак присваивания опущены, то результат вычисления присваивается переменной *ans* (от слова *answer* – ответ).

Имя переменной составляется из латинских букв, цифр и знака подчеркивания. Имя начинается с буквы. Прописные и строчные буквы различаются.

Операция ^ работает и с дробными показателями. Это позволяет вычислять корни, например, $\sqrt[5]{2^3} = 2^{3/5}$, выражение для *MATLAB*: 2^(3/5).

В выражениях можно использовать встроенные математические функции. Список элементарных математических функций выводится по команде **>> *help elfun***.

Пример. Вычислить значение функции $f(x) = \frac{1}{s\sqrt{2\pi}} e^{-\frac{(x-m)^2}{2s^2}}$ при $s = 0.5$, $m = 1.35$, $x = 0.1$.

Решение:

```
>> s = 0.5; m = 1.35; x = 0.1; 1/(s*sqrt(2*pi))*
   exp(-(x - m)^2/(2*s^2))
ans =
   0.0351
```

Если ввести переменную *f*, которой присваивается значение выражения 1/(*s***sqrt*(2**pi*))**exp*(–(*x* – *m*)^2/(2**s*^2)), то вместо *ans* = будет выведено *f* =.

Пример:

```
>> s = 0.5; m = 1.35; x = 0.1; f = 1/(s*sqrt(2*pi))*
   exp(-(x - m)^2/(2*s^2))
f =
   0.0351
```

1.2. Корректировка команд

После нажатия клавиши *Enter* команда выполняется и сохраняется в памяти (в стеке) и недоступна для корректировки, хотя она и присутствует на экране. Чтобы выполнить корректировку, необходимо вызвать нужную команду из стека в строку ввода клавишами ↑ или ↓ (стек можно листать повторным нажатием этих клавиш для поиска команды).

Поскольку команды и результаты всех предыдущих вычислений сохраняются, то последовательность вычислений можно организовать как последовательность нескольких команд.

Пример:

```
>> s = 0.5;
>> m = 1.35;
>> x = 0.1;
>> 1/(s*sqrt(2*pi))*exp(-(x - m)^2/(2*s^2))
ans =
   0.0351
```

Имеется также возможность длинную команду расположить на нескольких строках. Для продолжения команды на следующей строке необходимо после последнего символа, который сохраняется на данной строке, ввести три (или более) точки.

Пример:

```
>> s = 0.5;...
   x = 0.1;...
   m = 1.35;...
   1/(s*sqrt(2*pi))*...
   exp(-(x - m)^2/(2*s^2))
ans =
    0.0351
```

Операция продолжения команды на следующую строку особенно удобна при вводе матриц большой размерности, так как позволяет расположить матрицу на экране не в виде длинной строки, а в виде прямоугольной таблицы, как принято в математике.

1.3. Отмена вывода значения выражения

Точка с запятой в конце выражения отменяет вывод значения выражения на экран. Если команда состоит из нескольких выражений, то выражения отделяются друг от друга либо точкой с запятой, либо запятой. Запятая не отменяет вывода значения выражения, после которого она поставлена.

Пример:

```
>> s=0.5,m=1.35; x=0.1, f=1/(s*sqrt(2*pi))*exp(-(x-m)^2/(2*s^2))
s =
        0.5
x =
        0.1
f =
        0.0351
```

Не выводится $m = 1.35$, так как это выражение заканчивается символом «точка с запятой».

***Задача* 1.** *Вычислить* $b = y^{\sqrt[3]{(x-y)^4}} + \cos^3 y \dfrac{\left(\lg x + \ln y + \log_2 z\right)\left(1 + \dfrac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{|x-y|} + \dfrac{x}{2}}$ *при*

$x = 6.251$; $y = 0.827$; $z = 25.001$.

1.4. Выражения, содержащие матрицы

Матрицей размером $m \times n$ называется прямоугольная таблица чисел, имеющая m строк и n столбцов:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}.$$

Числа a_{ij} ($i = 1, 2, \dots, m$; $j = 1, 2, \dots, n$) называются элементами матрицы A. Первый индекс i указывает номер строки, второй индекс j – номер столбца, на пересечении которых расположен элемент a_{ij}. Если $m \neq n$, то матрица называется прямоугольной; если $m = n$, то матрица называется квадратной порядка n.

Одной из важнейших характеристик квадратной матрицы является ее определитель (детерминант), который обозначается в математике $|A|$ или $\det A$ составляется из тех же элементов (расположенных в том же порядке), что и матрица:

$$|A| = \det A = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & \dots & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{vmatrix}.$$

Определитель матрицы это число, которое получается из элементов матрицы по специальному правилу. Например, если $A=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, то $\det A = a_{11}a_{22} - a_{12}a_{21}$.

Для вычисления определителя матрицы A имеется функция ***det*** (A).

При вводе матрицы в командной строке элементы матрицы разделяются пробелом или запятой, строки разделяются точкой с запятой.

Пример. Ввести матрицу $A=\begin{bmatrix} 1.5 & -2.8 \\ 10 & 4 \end{bmatrix}$, вывести матрицу, вычислить и вывести ее определитель:

```
>> A = [1.5  -2.8;  10  4], D = det(A)
A =
    1.5  -2.8
     10  4
D =
    34
```

Прямоугольная матрица размером $1 \times n$ называется матрицей-строкой и ее элементы отмечаются одним индексом: $A = [a_1 \quad a_2 \quad \dots \quad a_n]$.

Матрица $m \times 1$ называется матрицей-столбцом: $B = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix}$.

Координаты вектора в математике принято записывать в виде матрицы-столбца, поэтому матрицу-столбец называют также вектором.

Квадратные матрицы n-го порядка вида

$$E=\begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix} \quad \text{и} \quad D=\begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & d_n \end{bmatrix}$$

называют соответственно единичной и диагональной матрицами.

Операции суммирования, вычитания, умножения и деления матрицы на число выполняются поэлементно.

Иначе выполняется операция умножения матриц. Умножение матрицы A на матрицу B определяется только при условии, что число столбцов матрицы A равно числу строк матрицы B, тогда элементы матрицы $C = AB$ вычисляются по правилу: элемент c_{ij} равен сумме произведений элементов i-й строки матрицы A на соответствующие элементы j-го столбца матрицы B. Из определения умножения матриц следует, что в общем случае $AB \neq BA$, т. е. результат произведения матриц изменяется при перестановке сомножителей.

Произведение матрицы-столбца на матрицу-строку всегда имеет смысл, а умножение матрицы-строки на матрицу-столбец возможно только при условии, что они содержат одинаковое количество элементов.

Матрица, полученная из матрицы A путем замены строк столбцами с сохранением их номеров, называется транспонированной по отношению к матрице A и обозначается в математике A' или A^T.

В *MATLAB* символ «штрих» после имени матрицы используется как команда «Выполнить операцию транспонирования данной матрицы».

Пример. Транспонировать матрицу $A = \begin{bmatrix} 1 & 2 \\ 3 & 1 \\ 2 & 4 \end{bmatrix}$. Вывести A и A':

```
>> A = [1 2; 3 1; 2 4], A'
A =
    1     2
    3     1
    2     4
ans =
    1     3     2
    2     1     4
```

Для транспонированной матрицы-строки будет матрица-столбец и наоборот.

Используя операцию транспонирования, вектор всегда можно записать как строку, например, вектор $B = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix}$ можно записать в виде $B = [b_1\ b_2\ \dots b_n]'$.

Матрицу, обратную квадратной матрице A, обозначают A^{-1} и определяют из условия $A\,A^{-1} = A^{-1}\,A = E$.

В *MATLAB* обратную матрицу вычисляет функция ***inv***(A). Обратную матрицу имеет только матрица, определитель которой $\det A \neq 0$.

Так как две любые квадратные матрицы одного и того же порядка можно перемножить, то можно найти матрицу $A \cdot A$. Эта матрица называется квадратом матрицы A и обозначается A^2. Аналогично определяется n-я степень A^n матрицы A. Нулевой степенью матрицы A называется единичная матрица E: $A^0 = E$.

Операции умножения, возведения в степень и деления матриц выполняются по специальным правилам, не поэлементно. Иногда эти операции требуется выполнить поэлементно, тогда применяются комбинированные знаки операций, которые состоят из двух символов: точка и знак обычной операции *, ^, /, \.

Пример. Выполним обычное и поэлементное умножение матриц

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \text{и} \qquad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}:$$

```
>> A = [1  2; 3  4]; B = [5  6; 7  8];
>> A * B
ans =
   19   22
   43   50
>> A. * B
ans =
   5    12
   21   32
```

Задача 2. *Ввести произвольную квадратную матрицу A четвертого порядка. Если* $\det A \neq 0$, *то*:

1) *проверить справедливость равенства* $A\,A^{-1} = A^{-1}\,A = E$;

2) *проверить справедливость равенства* $A^{-1} = \dfrac{A'}{\det A}$;

3) *возвести в квадрат матрицу A с использованием обычной операции возведения в степень и операции поэлементного возведения в степень*;

4) *ввести вектор* $B = \begin{bmatrix} 1.2 \\ -3.4 \\ 0.28 \\ -0.3 \end{bmatrix}$, *найти AB;*

5) *ввести матрицу-строку* C = [0.8 –0.7 3.7 1.2]. *Проверить, будет ли выполняться равенство* $BC = CB$.

1.5. Решение системы линейных алгебраических уравнений

Системой из m линейных уравнений с n неизвестными называется совокупность уравнений вида

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \qquad\qquad \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m. \end{cases}$$

Решением системы уравнений называется совокупность таких чисел $x_1 = \beta_1$, $x_2 = \beta_2$, …, $x_n = \beta_n$, которая обращает все уравнения системы в тождества.

Решение систем линейных алгебраических уравнений является наиболее часто встречающейся вычислительной задачей в прикладной математике.

Если система линейных уравнений имеет хотя бы одно решение, то она называется *совместной*, иначе – *несовместной*.

Совместная система, имеющая единственное решение, называется определенной, а система, имеющая более одного решения, – неопределенной.

Система линейных уравнений имеет очень компактную форму записи в матричном виде.

Составим матрицы:

$$A=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \quad X=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad B=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

тогда, используя правило умножения матрицы A на вектор X, можно записать равенство $AX = B$, это и есть матричная форма записи системы.

Рассмотрим важнейший частный случай системы, когда число уравнений равно числу неизвестных ($m = n$), тогда матрица A системы квадратная. Если $\det A \neq 0$, то система имеет единственное решение. Действительно, в этом случае матрица A имеет обратную матрицу A^{-1}. Умножим обе части равенства $AX = B$ слева на A^{-1}, получим $A^{-1}AX = A^{-1}B$. Но $A^{-1}A = E$, а $EX = X$, следовательно, $X = A^{-1}B$. Это и есть матричное решение системы линейных уравнений. Выполнив матричные операции в правой части, получим вектор X, координаты которого являются решением системы.

Отметим, что для системы линейных уравнений B – это вектор. Если B введена как строка, то формула решения системы имеет вид $X = A^{-1}B'$.

Пример. Решить систему уравнений $\begin{cases} 2x_1 - 3x_2 + 5x_3 = -1, \\ x_1 + x_2 + x_3 = 6, \\ 3x_1 + x_2 - 2x_3 = -1. \end{cases}$

Проверить решение. Действия по решению системы комментировать. Комментарий начинается символом **%**:

```
>> %ВВОД МАТРИЦЫ СИСТЕМЫ
>> A=[2  -3  5; 1  1  1; 3  1  -2]
A =
    2    -3     5
    1     1     1
    3     1    -2
>> %ВЫЧИСЛЕНИЕ ОПРЕДЕЛИТЕЛЯ МАТРИЦЫ СИСТЕМЫ
>> det(A)
ans =
   -31
>> %СИСТЕМА ИМЕЕТ ЕДИНСТВЕННОЕ РЕШЕНИЕ
>> %ВВОД  ВЕКТОРА ПРАВОЙ ЧАСТИ СИСТЕМЫ
>> B=[ -1;  6;  -1]';
>> % РЕШЕНИЕ СИСТЕМЫ МАТРИЧНЫМ МЕТОДОМ
>> X = inv(A)*B
X =
   -0.1613
```

```
    3.9355
    2.2258
>> %ПРОВЕРКА РЕШЕНИЯ
>> A*X
ans =
    -1
     6
    -1
>> %РАВЕНСТВО  AX = B  ВЫПОЛНЯЕТСЯ
```

Эту же систему можно решить с помощью команды, в которой используется операция \ левого деления матриц: $X = A \setminus B$.

***Задача* 3**:

1) *решить систему, используя операцию левого деления*

$$\begin{cases} 2,1546x_1 + 0,8431x_2 + 0,3146x_3 + 0,1615x_4 = 3,1826, \\ 0,8431x_1 + 3,1415x_2 + 0,6241x_3 + 0,2131x_4 = 4,6123, \\ 0,3146x_1 + 0,6241x_2 + 4,8216x_3 + 0,8245x_4 = 5,9681, \\ 0,1615x_1 + 0,2131x_2 + 0,8245x_3 + 6,4131x_4 = 8,1418; \end{cases}$$

2) *решить систему по формуле* $X = A \setminus B$

$$\begin{cases} 3,81x_1 + 0,25x_2 + 1,28x_3 + (0,75 + a)x_4 = 4,21, \\ 2,25x_1 + 1,32x_2 + (4,58 + a)x_3 + 0,49x_4 = 6,47 + b, \\ 5,31x_1 + (6,28 + a)x_2 + 0,98x_3 + 1,04x_4 = 2,38, \\ (9,39 + a)x_1 + 2,46x_2 + 3,35x_3 + 2,28x_4 = 10,48 + b, \end{cases}$$

где $a = 1,5$, $b = 0,25$.

1.6. Произвольные системы линейных уравнений

Рассмотрим системы, у которых число неизвестных не равно числу уравнений. Матрицу такой системы называют прямоугольной.

Наивысший порядок отличных от нуля миноров прямоугольной матрицы A называется ее рангом и обозначается символом *rank A*. В *MATLAB* ранг матрицы A вычисляет функция ***rank***(A).

Необходимым и достаточным условием совместности произвольной системы линейных уравнений является равенство рангов матрицы системы и ее расширенной матрицы. Расширенная матрица получается добавлением к матрице системы столбца свободных членов.

Если число уравнений системы меньше числа неизвестных, то система называется *недоопределенной*.

***Задача* 4.** *Проверить совместность системы. Если система совместна, решить ее по формуле* $X = A \setminus B$

$$\begin{cases} 5x_1 + 3x_2 + 2x_3 + 6x_4 + 4x_5 = -6, \\ 3x_1 + 5x_2 + 2x_3 + 5x_4 + 7x_5 = -19, \\ 3x_1 + 9x_2 + 3x_3 + 10x_4 + 8x_5 = -24, \\ 10x_1 + 2x_2 + 3x_3 + 7x_4 + 7x_5 = -7. \end{cases}$$

*Можно ли эту систему решить по формуле X = inv (A)*B ?*

Если число уравнений системы больше числа неизвестных, то система называется *переопределенной*.

***Задача* 5.** *Проверить совместность системы. Если система совместна, решить ее по формуле X=A \ B*

$$\begin{cases} 0.5x_1 + x_2 = 1, \\ 2x_1 + x_2 = 0.25, \\ 3x_1 + 2x_2 = 1. \end{cases}$$

*Можно ли эту систему решить по формуле X = inv (A)*B ?*

1.7. Вычисление собственных значений и собственных векторов матрицы

Большое число задач механики, физики и техники требуют отыскания собственных значений и собственных векторов матриц.

Собственным значением квадратной матрицы *A* называется такое значение величины λ, при котором система уравнений $AX = \lambda X$ имеет ненулевое решение. Это решение называется собственным вектором матрицы *A*, соответствующим данному собственному значению λ.

Собственные значения и собственные векторы матрицы *A* определяет команда

```
>> [U, D] = eig(A)
```

Диагональная матрица *D* содержит собственные числа, а столбцы матрицы *U* являются собственными векторами матрицы *A*.

Пример. Найти собственные числа и собственные векторы матрицы

$$A = \begin{bmatrix} 2 & -3 & 5 \\ 1 & 1 & 1 \\ 3 & 1 & -2 \end{bmatrix}.$$

Решение:

```
>> [U, D] = eig(A)
U =
   -0.6224    0.6996   -0.3970
   -0.0294    0.5172   -0.7828
    0.7821    0.4930   -0.4792
```

```
D =
   -4.4250         0         0
         0    3.3058         0
         0         0    2.1192
```

***Задача* 6.** *Найти собственные числа и собственные векторы матрицы*

$$A=\begin{bmatrix} 3.8100 & 0.2500 & 1.2800 & 0.7500 \\ 2.2500 & 1.3200 & 4.5800 & 0.4900 \\ 5.3100 & 6.2800 & 0.9800 & 1.0400 \\ 9.3900 & 2.4600 & 3.3500 & 2.2800 \end{bmatrix}.$$

2. ГРАФИКА *MATLAB*

График дает самую полную общую информацию о функции, поэтому в системах компьютерной математики большое внимание уделяется средствам построения графиков функций.

В *MATLAB* имеется три функции построения двумерных графиков: ***plot***(), ***fplot***() и ***ezplot***().

2.1. Функция *plot*()

Функция ***plot***() строит графики кривых с массивами значений абсцисс x и ординат y и имеет следующий формат:

```
>> plot(x, y, 's')
```

Строка s используется для задания параметров графика: тип, толщину и цвет рисуемой линии, а также форму и размер меток на графике. Строка s может отсутствовать, тогда *MATLAB* устанавливает параметры, действующие по умолчанию: все кривые выводятся сплошными линиями, окрашенными циклически в шесть различных цветов.

Переменная s представляет собой строку символов, и эту строку можно подготовить в отдельной команде или записать непосредственно в команде построения графика, например,

```
>> s = 'k -.';
>> plot(x, y, s)
```

или

```
>> plot(x, y, 'k-.')
```

Таблица кодировки типов линий при выводе графика функции

Тип линии	***Код***
Сплошная	**–** (знак минус)
Пунктирная	**:** (двоеточие)
Штрихпунктирная	**–.** (минус и точка)

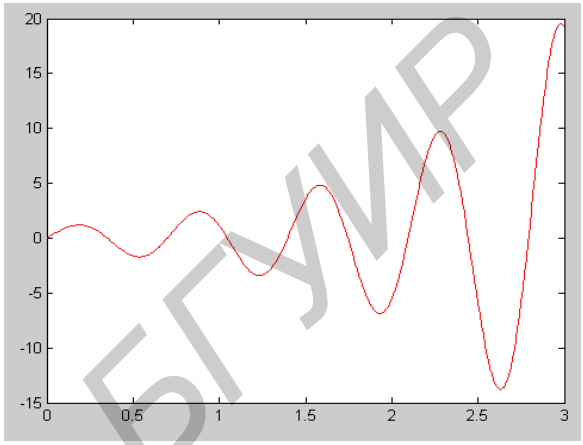
Таблица кодировки цвета линии графика функции

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

Таблица кодировки меток на графике

Описание	Точка	Звезда	Кружок	Квадрат	Плюс	Крест	Ромб	Треугольники
Код	**.**	*****	*Буква* ***o***	***s***	***p***	***x***	***d***	**^ *v* > <**

Пример. Построить график функции $y = e^x \sin(9x)$ на промежутке [0; 3] с шагом 0.01. График рисовать сплошной линией красного цвета.



Для вычисления массива значений *y* должны использоваться посимвольные операции с матрицами.

Задачу построения графика решаем с помощью следующей команды:

```
>> x = 0:0.01:3; y = exp(x)*
   sin(9*x); plot(x, y,'r-')
```

Размер метки можно задать параметром *MarkerSize* в команде *plot*(), например,

```
>> x=0:0.01:3; y=exp(x)*sin(9*x); plot(x,y,'k:o','MarkerSize',10)
```

Для построения графиков используется специальный графический редактор, который строит на экране графическое окно и в нем рисует график. Вызывается графический редактор автоматически при выполнении графических функций. По умолчанию производится разметка осей координат и иногда выводится заголовок сверху графика. Графический редактор имеет свои средства модификации графика и нанесения на график различных надписей и символов. Например, заголовок создает функция ***title***(‘*text*’), причем в качестве параметра *text* можно использовать и формулу.

Пример. Построить график функции $y = e^x \sin(9x)$ с заголовком:

```
>> x=0:0.01:3; y=exp(x)*sin(9*x);
   plot(x,y,'r-'), title('y=exp(x)sin(9x)')
```

Текст можно вывести в заданное координатами (*x*, *y*) место графика функцией ***text*** (*x*, *y*, *'TEXT'*).

Пример:

```
>> x=0:0.01:3;  y=exp(x).*sin(9*x);   plot(x, y,'r-')
>> text(0.5, 15, 'grafik y=exp(x)sin(9x)')
```

Координаты вывода текста можно указать на графике мышью, если использовать функцию ***gtext*** (*'TEXT'*). Тогда после вывода графика и выполнения команды

```
>> gtext('text')
```

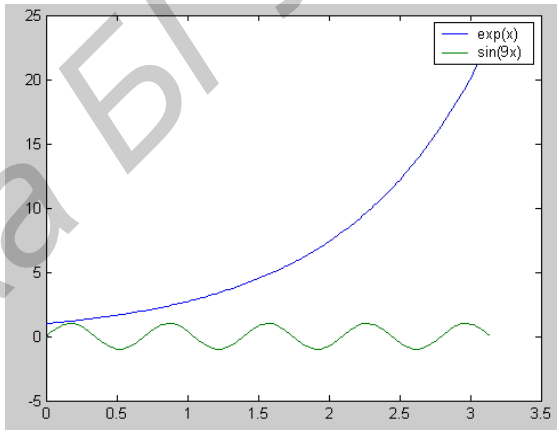
нужно указать мышью место на графике и щелкнуть.

Пример:

```
>> x=0:0.01:3;  y=exp(x).*sin(9*x);   plot(x, y,'r-')
>> gtext('grafik y=exp(x)sin(9x)')
```

На одном чертеже можно построить графики нескольких функций, для этого нужно попарно перечислить векторы, задающие абсциссы и ординаты. Для идентификации графиков можно использовать команду ***legend***(), позволяющую связать с каждой кривой некоторую текстовую информацию.

Пример. Построить на одном чертеже графики функций $y = e^x$ и $y = \sin(9x)$.



Решение:

```
>> x=0:0.02:pi; y=exp(x);
   z=sin(9*x); plot(x,y,x,z),
   legend('exp(x)', 'sin(9x)')
```

Имеется другой способ размещения на одном чертеже графиков нескольких функций: сформировать матрицу, столбцы которой должны содержать нужные ординаты. Ординаты обычно удобно задавать строками, тогда для построения графиков строки необходимо транспонировать.

Пример. Построить на одном чертеже графики функций $\sin 2x$, $\sin 3x$, $\sin 4x$:

```
>> x=0:0.02:pi; Y=[sin(2*x)' sin(3*x)' sin(4*x)'];plot(x,Y)
```

Задача 7:

1) *построить график функции* $y = e^x \sin(9x)$ *на промежутке* $[0; 3]$ *с шагом* 0.01. *График рисовать сплошной линией красного цвета*;

2) *построить график функции* $y = e^x \sin(9x)$ *на промежутке* $[0; 3]$ *с шагом* 0.01. *График рисовать пунктирной линией черного цвета с маркерами в виде пятиконечной звезды размером* 5;

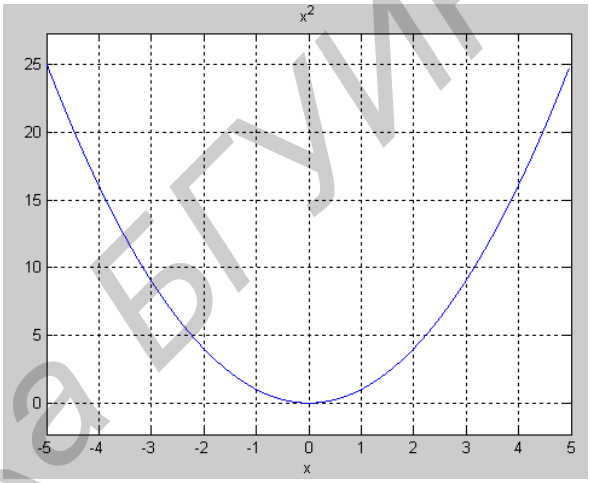
3) *построить на одном чертеже графики функций* $y = 20e^{-(x-5)^2}$, $z = -x^2 + 9x - 8$, $x \in [0;12]$, $h = 0.1$;

4) *построить на одном чертеже графики функций* $y = 20e^{-(x-5)^2}$, $z = -x^2 + 9x - 8$, $x \in [0;12]$, $h = 0.1$. *Использовать функцию legend для идентификации графиков*;

5) *построить на одном чертеже графики функций* $y = 2\sin x^3$, $y = 10xe^{-x}$, $y = \sqrt[4]{x^2 + 2}$, $x \in [0; \pi]$, $h = 0.02$. *Использовать функцию legend.*

2.2. Функция *ezplot*()

Функция ***ezplot***() применяется для рисования графиков неявно заданных функций двух переменных и параметрически заданных функций. Эта функция рисует также графики функций одной переменной, причем, в отличие от функции *plot*() шаг и нужные массивы значений абсцисс и ординат создаются автоматически. Функция *ezplot*() автоматически выводит над графиками наименование функций.



Пример. Построить на промежутке [–5; 5] график функции $y = x^2$. Нанести на график сетку.

Решение:

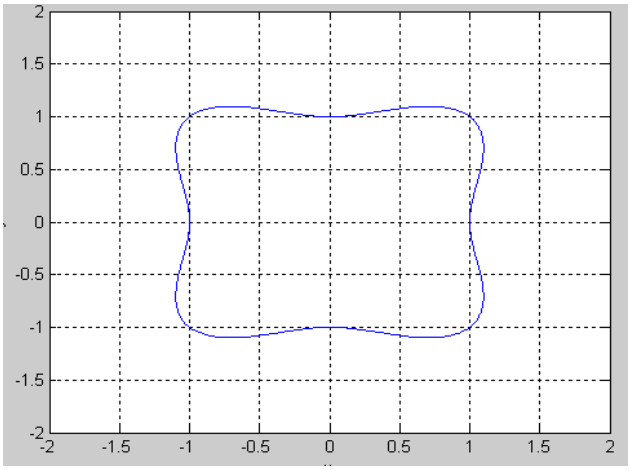
```
>> ezplot('x^2',[ -5 5]), grid on
```

Пример. Построить на промежутке [–2; 2] график неявной функции, заданной уравнением $x^2 + y^2 = x^4 + y^4$.

Решение:

```
>> ezplot('x^2+y^2-x^4-y^4',[ -2,2]), grid
```

Функция *ezplot*() особенно удобна для определения начального приближения решения системы двух уравнений $\begin{cases} f_1(x, y) = 0, \\ f_2(x, y) = 0 \end{cases}$ графическим методом. Нужно построить на одном чертеже графики двух неявных функций $f_1(x, y) = 0$ и $f_2(x, y) = 0$. Эту задачу решают две последовательно выполненные команды

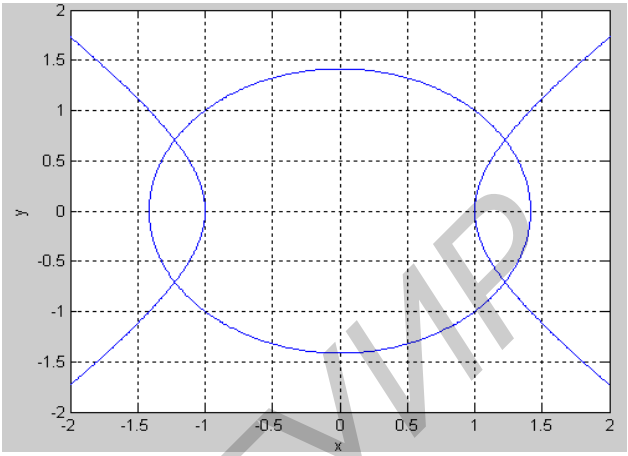



```
>> ezplot('f1(x,y)'), hold
>> ezplot('f2(x,y)'), grid
```

В первой команде используется функция ***hold***, чтобы сохранить рисунок для последующего наложения на него второго рисунка.

Пример. Найти начальное приближение решения системы

$$\begin{cases} 2 - x^2 - y^2 = 0, \\ 1 - x^2 + y^2 = 0. \end{cases}$$



Решение:

```
>> ezplot('2–x^2–y^2',[–2,2]), hold
>> ezplot('1–x^2+y^2',[–2,2]), grid
```

Отметим, что первое уравнение $2 - x^2 - y^2 = 0$ является уравнением окружности, а при выводе окружность сжимается и приобретает вид эллипса. На значение начального приближения это не влияет, так как в таком же масштабе сжимается и второй график. Но вывести, например, первый график можно неискаженным, если потребовать использовать равное масштабирование по обоим осям функцией ***axis equal***:

```
>> ezplot('2–x^2–y^2',[ –2,2]), axis equal, hold
```

Рисунок показывает, что система имеет четыре решения, приближенные значения которых (1.25; 0.75), (1.25; –0.75), (–1.25; 0.75), (–1.25; –0.75).

***Задача* 8.** *Найти графическим методом начальные приближения решений системы* (*N* − *номер варианта*):

$$\begin{cases} (1 + N/10)x^3 - y^2 - 1 = 0, \\ xy^3 - y - 4 = 0. \end{cases}$$

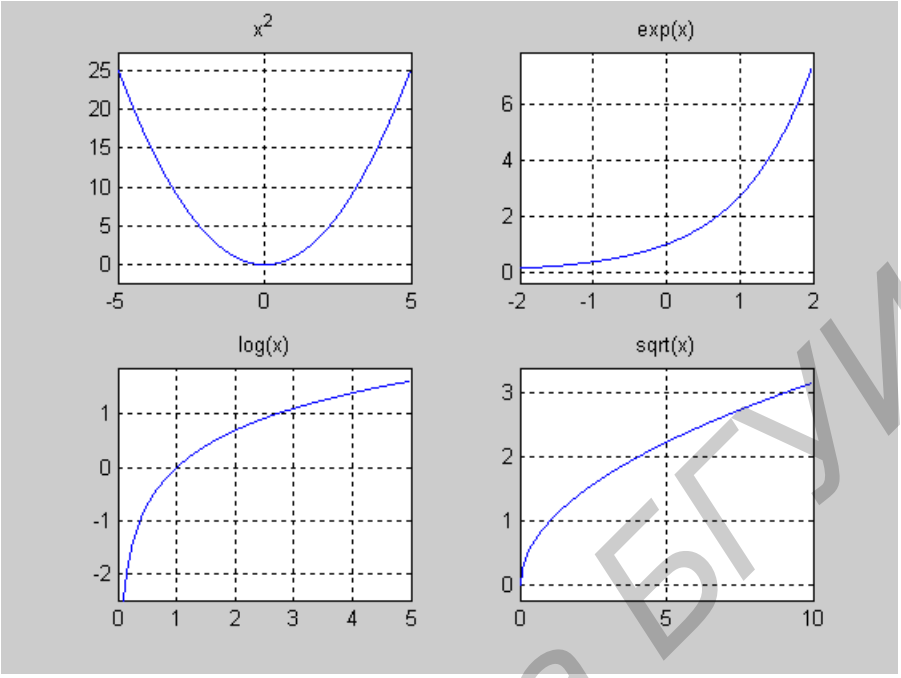
2.3. Построение нескольких рисунков в одном графическом окне

В одном графическом окне можно разместить несколько рисунков, используя функцию ***subplot***(*N, M, K*). Эта функция создает массив графиков, состоящий из *N* рисунков по вертикали и *M* рисунков по горизонтали. В результате выполнения функции *subplot*(*N, M, K*) активным становится рисунок с номером *K*. Нумерация ведется слева направо и сверху вниз. Крайний слева рисунок из верхнего ряда считается первым, а крайний справа из нижнего ряда имеет номер *N***M*.

Пример. В одном графическом окне построить графики четырех функций: $y = x^2$ на промежутке [–5; 5], $y = e^x$ на промежутке [–2; 2], $y = \lg x$ на промежутке [0; 5] и $y = \sqrt{x}$ на промежутке [0; 10].

Решение. Выполним следующие команды, не закрывая графического окна:

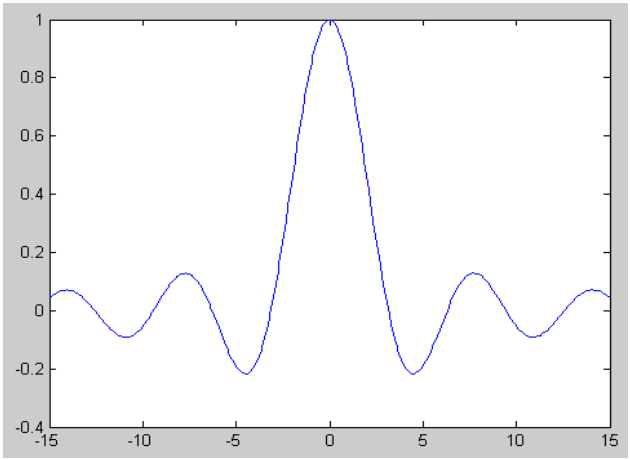
```
>> subplot(2,2,1), ezplot('x^2',[ −5,5]), xlabel(''), grid
>> subplot(2,2,2), ezplot('exp(x)',[ −2,2]), xlabel(''), grid
>> subplot(2,2,3), ezplot('log(x)',[0,5]), xlabel(''), grid
>> subplot(2,2,4), ezplot('sqrt(x)',[0,10]), xlabel(''), grid
```



В командах вывода графиков использована функция ***xlabel*** (' ') с аргументом в виде пустой строки, которая отменяет вывод обозначения горизонтальной оси. Если этого не сделать, то символ обозначения оси наложится на наименование функций нижних графиков.

2.4. Функция *fplot*()

Функция ***fplot***('*f*(*x*)', [*x*min, *x*max]), аналогично функции *plot*(), строит график функции *f*(*x*), однако при использовании функции *fplot*() не требуется задавать массив значений аргумента и вычислять массив значений функции, все это делается автоматически. График строится на промежутке изменения аргумента *x* от *x*min до *x*max.



Пример. Построить график функции $y = \frac{\sin x}{x}$ на промежутке [–15; 15].

Решение. Используем функцию *fplot*():

```
>> fplot('sin(x)/x',[ −15,15])
```

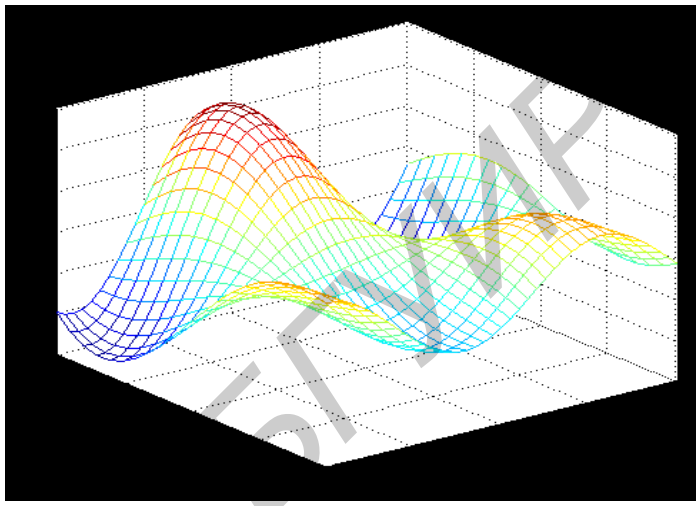
2.5. Трехмерная графика

Чтобы построить поверхность $z = f(x, y)$, нужно иметь массив значений функции z, вычисленный на некоторой сетке. Для формирования двумерной прямоугольной сетки используется функция ***meshgrid***(x, y), где x и y − одномерные массивы точек, задающие абсциссы и ординаты двумерной функции.

Пример. Построить поверхность $z = 2\cos(x+y) + y\cos(x-y)$.

Решение. Для построения поверхности используем функцию *mesh* (*X, Y, Z*):

```
>> x=0:0.2:8; y=0:0.2:4;
   [X,Y] = meshgrid(x,y);
   Z=2*cos(X+Y)+Y.*cos(X-Y);
   mesh(X,Y,Z)
```



Функция ***mesh***() строит расцвеченную сетчатую поверхность, используя различную окраску вершин и ребер.

Кроме функции *mesh*(), для изображения поверхности имеются и другие функции:

meshc() − строится сетчатая поверхность с линиями уровня;
meshz() − строится сетчатая поверхность и отсчетная плоскость;
surf() − строится расцвеченная поверхность;
surfc() − строится расцвеченная поверхность с линиями уровня;
surfl() − строится расцвеченная поверхность с подсветкой;
waterfall − строится поверхность без прорисовки ребер.

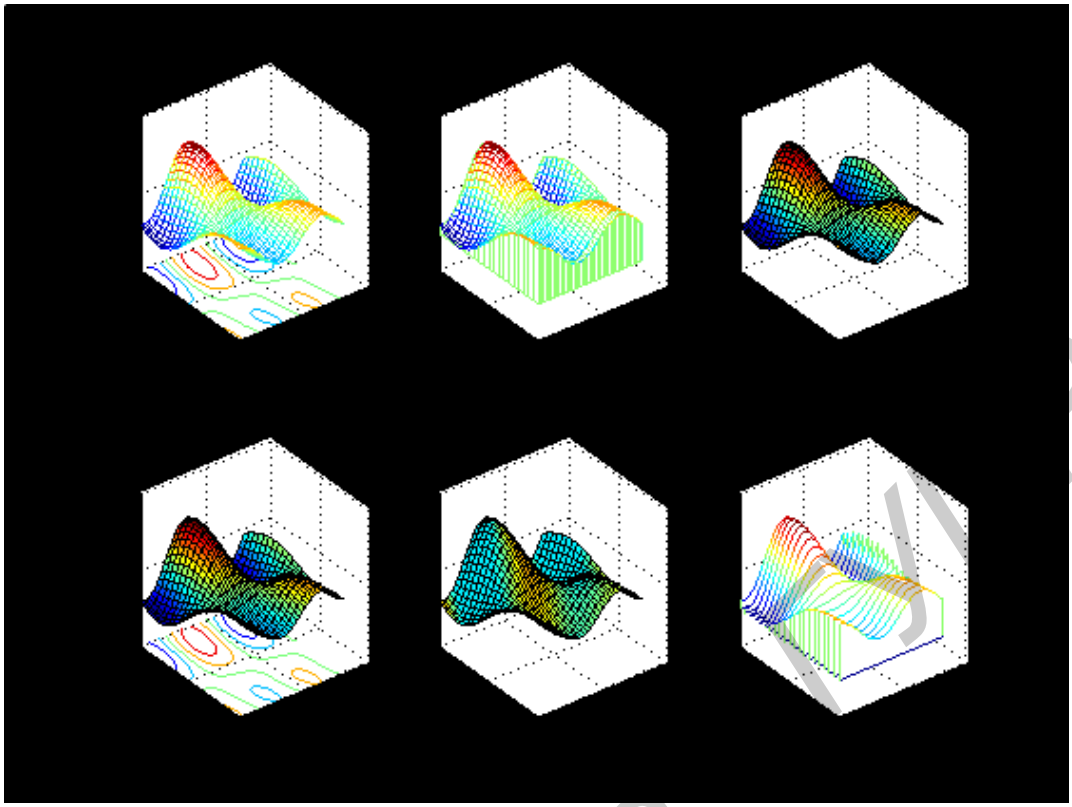
Пример. Построим поверхность $z = 2\cos(x+y) + y\cos(x-y)$ этими функциями:

```
>> x=0:.2:8; y=0:.2:4; [X,Y]=meshgrid(x,y);
   Z=2*cos(X+Y)+Y.*cos(X-Y); X-Y);
>> subplot(2,3,1); meshc(X,Y,Z)
>> subplot(2,3,2); meshz(X,Y,Z)
>> subplot(2,3,3); surf(X,Y,Z)
>> subplot(2,3,4); surfc(X,Y,Z)
>> subplot(2,3,5); surfl(X,Y,Z)
>> subplot(2,3,6); waterfall(X,Y,Z)
```

После построения рисунков для каждого рисунка можно выполнить масштабирование функцией ***axis*** ([*−Inf Inf −Inf Inf −Inf Inf*]). Например, второй рисунок масштабируем с помощью команды

```
>> subplot(2,3,2); meshc(X,Y,Z), axis([-Inf Inf -Inf Inf -Inf Inf])
```

При масштабировании определяются действительные интервалы изменения величин по всем координатам, и рисунки становятся выразительнее.



***Задача* 9**:

1) *поверхность* $z = \operatorname{arctg}\dfrac{x+y}{1-xy}$ *построить с помощью функции mesh*() *на сетке* $x = 0{:}0.2{:}8;\ y = 0{:}0.2{:}4$. *Использовать режим вращения изображения для изменения ракурса. Режим вращения задается нажатием соответствующего значка на панели инструментов графического окна*;

2) *поверхность* $z = \operatorname{arctg}\dfrac{x+y}{1-xy}$ *построить на сетке* $x = 0{:}0.05{:}8;\ y = 0{:}0.05{:}4;$

3) *поверхность* $z = \operatorname{arctg}\dfrac{x+y}{1-xy}$ *построить на сетке* $x = 0{:}0.02{:}8;\ y = 0{:}0.01{:}6;$

4) *построить поверхность* $z = \sqrt{xy}$ *функцией meshc*() *на сетке* $x = 0{:}0.2{:}8;\ y = 0{:}0.1{:}6;$

5) *выбрать сетку и построить поверхность* $z = x^y e^{-x}$ *функцией meshz*();

6) *построить в одном графическом окне три поверхности* $z = e^x \sin(y)$: *одну на сетке* $x = 0{:}0.1{:}8;\ y = 0{:}0.1{:}4$, *вторую на сетке* $x = 0{:}0.05{:}8;\ y = 0{:}0.05{:}4$ *и третью на сетке* $x = 0{:}0.01{:}8;\ y = 0{:}0.01{:}4;$

7) *выбрать сетку и построить в одном графическом окне четыре поверхности* $z = e^{ax+by}$ *при следующих значениях параметров a и b:* $a = 1, b = -1;\ a = -1, b = 1;\ a = 5, b = 5;\ a = -5, b = -5;$

8) *построить в одном окне поверхность* $z = x\sin 2y + y\cos 3x$ *функциями meshz*() *и surf*() *на сетке x = –pi*:0.1:*pi; y = –pi*:0.1:*pi. Выполнить масштабирование рисунков*;

9) *построить поверхность* $z = \sin(x)\sin(3y)$ *функцией waterfall*() *на сетке x = –pi*:0.1:*pi; y = –pi*:0.1:*pi. Установить единый масштаб по осям, выбрать и установить ракурс, удалить сетку.*

3. ФОРМАТИРОВАНИЕ ВЫВОДА

Формат вывода по умолчанию часто оказывается не лучшим. Имеются функции, позволяющие организовать вывод в удобной для пользователя форме.

Для вывода только текста можно использовать функцию ***disp*** (*'Текст'*).

Форматированный вывод результатов вычислений (и текста) выполняют функции *fprintf* () и *sprintf* (). Функция ***fprintf*** () используется в виде

```
fprintf('строка символов', список выводимых переменных)
```

В строке символов можно записывать последовательность любых символов, в том числе и пробелы. Символы записываются сплошной строкой, без разделителей. Два символа \ и **%** в строке имеют специальное назначение, они зарезервированы для обозначения команд управления выводом.

Команды, начинающиеся символом \, состоят из двух символов и означают следующее:

n − переход на новую строку;

t − горизонтальная табуляция;

b − возврат на один символ;

r − возврат в начало строки;

f − переход на новую страницу;

\\ − вывод символа \;

? − вывод символа **?**.

Соседство символа \ с иными символами справа в строке символов недопустимо.

Пример:

```
>> x=5; y=7; disp('Вывод значений x и y'); fprintf('x=%d y=%d',x,y)
Вывод значений x и y
x=5   y=7
```

***Задача* 10.** *Выполнить указанные команды и объяснить различное представление* (*или не представление*) *результатов вывода*:

```
>> x=7; y=6;
```

```
>> fprintf('x=%d    y=%d',x, y)
>> fprintf('\n%d',x, y)
>> fprintf('x=%d   \ny=%d',x, y)
>> fprintf('\%d',x, y)
>> fprintf('\?%d',x, y)
>> fprintf('%d',x, y)
>> fprintf('\\%d\n',x, y)
>> fprintf('\d%d\n',x, y)
```

Команды, начинающиеся символом **%**, состоят в зависимости от назначения из различного количества символов справа. Заканчиваются эти команды одной из следующих букв:

c − вывод одиночного символа;
d − вывод целого числа;
e − экспоненциальное представление числа, например 3.1415*e*+00;
E − экспоненциальное представление числа, например 3.1415*E*+00;
f − действительное число с фиксированной точкой;
g − наиболее компактный вариант из *e* и *f*; незначащие нули не выводятся;
G − то же, что и ***g***, но вместо «*e*» используется «*E*»;
o − число в восьмеричной системе счисления;
u − десятичное число без знака;
s − вывод строки символов;
x − число шестнадцатеричной системы с использованием символов нижнего регистра;
X − то же, что и *x*, но с использованием символов верхнего регистра.

Между символом процента и буквой конца команды можно вставить дополнительные символы, список которых приводится в таблице

Символ	***Описание***	***Пример***
– (знак минус)	Выравнивает значение по левому краю	%–5.2f
+ (знак плюс)	Выводить число со знаком	%+5.2f
0 (ноль)	Заполнение нулями вместо пробелов	%0 5.2f
Целое число	Определяет минимальное число знаков, которые будут выведены	%5f
. целое число	Число после точки определяет количество символов, выводимых справа от десятичной точки	%5.2f

Последовательность символов, которая не начинается символом \ или **%**, рассматривается как комментарий и выводится в виде текста.

***Задача* 11**:

1) *выполнить последовательность команд*:

```
n = 15;
err = 128.3567;
value = 1234567890;
fprintf('Ответ:')
fprintf('\nШаг %d, Ошибка %.3f \nx=%–15.3e ', n, err, value)
```

Изменить команды, чтобы вывод занимал одну строку.

2) *ввести матрицу второго порядка A. Для форматированного вывода матрицы и ее определителя выполнить команды*

```
>> A=[1.5 –2.8;10 4];
>> disp('Матрица A'), fprintf('%2.2f  %2.2f',A(1,1),A(1,2)),
   fprintf('\n%2.2f  %2.2f',A(2,1),A(2,2))
```

4. ОСНОВЫ ПРОГРАММИРОВАНИЯ В *MATLAB*

4.1. Создание и выполнение программы

Для простых операций удобен командный режим, но если вычисления нужно многократно повторять или необходимо реализовать сложные алгоритмы, то создается и записывается в файл программа вычислений.

Система *MATLAB* имеет свой язык программирования высокого уровня.

Для создания программы нужно вызвать специальный редактор файлов, выполнив команду *File* / *New* / *M-file*, которая выводит окно редактора файлов со своим меню. В окне редактора набирается текст программы, программу следует сохранить, указав имя файла, затем закрыть редактор.

Файлы, созданные редактором файлов *MATLAB*, называются *M-файлами.*

Чтобы выполнить программу, в командной строке указывается имя соответствующего файла и нажимается *Enter*.

Если в программе есть ошибки, то нужно вывести текст программы, выполнив команду *File* / *Open* и выбрав нужный файл. Исправить ошибки, сохранить исправленную программу и закрыть окно редактора. Программу запустить на выполнение.

Пример. Треугольник задан координатами вершин $A(0;\ 0)$, $B(i;\ i-1)$, $C(-i;\ i+1)$, где i – номер варианта. Создать программу вычисления длины медианы, проведенной к стороне a, длины биссектрисы угла A, площади треугольника и радиуса описанной окружности.

Решение. Возьмем номер варианта $i = 0$. Длины сторон треугольника вычисляем как расстояния между вершинами по формуле $d=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$, длину медианы, проведенной к стороне a, вычисляем по формуле $m_a=\dfrac{\sqrt{2(b^2+c^2)-a^2}}{2}$, длину биссектрисы угла A вычисляем по формуле $l_A=\dfrac{\sqrt{bc(b+c)^2-a^2}}{b+c}$, площадь треугольника вычисляем по формуле $S=\sqrt{p(p-a)(p-b)(p-c)}$, $p=(a+b+c)/2$, радиус описанной окружности вычисляем по формуле $R=\dfrac{abc}{4S}$.

Листинг программы:

```
i=0;
A=[0 0]; B=[i i–1]; C=[–i i+1];
a=sqrt((B(2) –B(1))^2+(C(2) –C(1))^2)
b=sqrt((A(2) –A(1))^2+(C(2) –C(1))^2)
c=sqrt((A(2) –A(1))^2+(B(2) –B(1))^2)
ma=sqrt(2*(b^2+c^2) –a^2)/2
lA=sqrt(b*c*((b+c)^2–a^2))/(b+c)
p=(a+b+c)/2; S=sqrt(p*(p–a)*(p–b)*(p–c))
R=(a*b*c)/(4*S)
```

***Задача* 12.** *Треугольник задан координатами вершин* $A(0; 0)$, $B(N; N-1)$, $C(-N; N+1)$, *где N – номер варианта. Составить программу вычисления длины медианы, проведенной к стороне b, длины биссектрисы угла C, площади треугольника и радиуса описанной окружности*.

4.2. Оператор *if*

Простейшее предложение с оператором ***if*** имеет вид

```
f  BOOL, EXPR, end
```

где *BOOL* – условие, *EXPR* – команды.

Команды *EXPR* будут выполнены, если условие *BOOL* истинно.

Пример. Составить программу, которая вычисляет корень четвертой степени, если значение выражения положительно, и выводит сообщение, если оно отрицательно. Сообщение выводит оператор ***disp*** (*'Текст'*):

```
x=5;
y=(x*cos(x) –3);
if y>=0, z=y^(1/4), end
if y<0, disp('y<0, не существует действительное значение z'),end
```

В предложение с оператором *if* можно включить альтернативу с помощью оператора ***else***:

```
x=5;
y=(x*cos(x) –3);
if y>=0, z=y^(1/4)
else disp('y<0, не существует действительное значение z'), end
```

Предложение с оператором *if* будет мощнее, если использовать дополнительный оператор ***elseif***:

```
if  BOOL_1,  EXPR_1
elseif  BOOL_2,  EXPR_2
elseif  BOOL_3,  EXPR_3
   ...
else  EXPR_0
end
```

Здесь проверки следуют одна за другой. В том случае, если условие *BOOL_N* истинно, будут выполнены команды *EXPR_N*.

Пример. В результате выполнения программы

```
x=3;
if x<0, x=–x
elseif x<3, x=x^2
  elseif x<5, x=sqrt(x)
      elseif x<7, x=sin(x)
      else x>10, x=log(x)
      end
```

будет получен ответ $x = 1.7321$, так как выполняется команда `x=sqrt(x)`.

В условных операторах используются операции отношения и логические операции. Имеются следующие операции отношения: **<**, **<=**, **>**, **>=**, **==** (равно), **~=** (не равно); и логические операции: **&** (логическое «И»), **|** (логическое «ИЛИ»), **~** (отрицание – логическое «НЕ»).

4.3. Оператор-переключатель *switch*

Предложение с оператором ***switch*** имеет вид

```
switch VAR
case VAR1, EXPR_1
case {VAR2, VAR3, …}, EXPR_2
otherwise, EXPR_0
end
```

Здесь *VAR1*, *VAR2*, *VAR3*, … – различные значения, которые может принимать переменная *VAR*, а *EXPR_1*, *EXPR_2* и т. д. – группы команд.

Переключение возможно как по единственному значению (выбор *VAR1*), так и по группе значений (выбор из *VAR2*, *VAR3*, …). Кроме того, если выбор не сделан, то выполняются команды после оператора ***otherwise*** (команды *EXPR_0*).

Пример:

```
x=3.5;
switch x
      case 2, y=x^2
           case {3, 4, 5}, y=exp(x)
           otherwise, y=0
 end
```

***Задача* 13.** *Составить программу, которая выводит текст 'Успеваемость низкая', если имеются оценки* 1, 2 *или* 3; *'Успеваемость средняя', если имеются оценки* 4, 5, 6 *или* 7; *'Успеваемость высокая', если имеются оценки* 8 *или* 9; *'Успеваемость отличная' при оценках* 10.

4.4. Оператор цикла *for*

Предложение с оператором ***for*** имеет вид

```
for N=N0:DN:N1, EXPR, end
```

Здесь N – счетчик, пробегающий значения от $N0$ до $N1$ с шагом DN (учитываются вещественные части чисел), а $EXPR$ обозначает выполняемые в теле цикла команды.

Пример. Составить программу вычисления таблицы значений функции $y = x + \dfrac{1}{2x^2 + \dfrac{1}{3x^3 + \dfrac{1}{4x^4}}}$ на промежутке [0.5; 1.5] с шагом 0.1.

Программа:

```
disp('Таблица функции')
for x=0.5:0.1:1.5
    y=x+1/(2*x^2+1/(3*x^3+1/(4*x^4)));
    fprintf('x=%1.1f   y=%1.3f\n',x,y),
end
```

Результат выполнения программы:

```
Таблица функции
x=0.5   y=1.873
x=0.6   y=1.502
x=0.7   y=1.384
x=0.8   y=1.373
x=0.9   y=1.398
x=1.0   y=1.433
x=1.1   y=1.476
x=1.2   y=1.526
x=1.3   y=1.583
x=1.4   y=1.647
x=1.5   y=1.717
```

***Задача* 14.** *Составить программу вычисления таблицы значений функции* $f(x) = \sin(x^3 + 31.2x - N) - e^{-0.001x} + \dfrac{1}{x^2 + 3}$ *на промежутке* [−1.5; 1.5] *с шагом* 0.2, N *– номер варианта*.

4.5. Оператор цикла *while*

Цикл с оператором ***while*** имеет вид

```
while  BOOL, EXPR, end
```

Цикл выполняется до тех пор, пока условие *BOOL* истинно.

Для досрочного выхода из тела цикла можно использовать оператор прерывания ***break***. Имеется также оператор ***continue*** для пропуска команд и перехода к следующему значению переменной цикла.

Пример. Составить программу вычисления таблицы неотрицательных значений функции $y = \frac{\sqrt{e^x + tgx + 4}}{\sin^3(x^2 + 2.5)} - \lg(\cos x^3 + 3.654x^4 + 5)$, начальное значение $x = 0$, шаг таблицы 0.05.

Программа:

```
x=0; y=1;
disp('Таблица функции')
while y>0
   y=sqrt(exp(x)+tan(x)+4)/(sin(x^2+2.5))^3-log10(cos(x^3)
     +3.654*x^4+5);
   fprintf('x=%1.2f   y=%1.3f\n',x,y)
   x=x+0.05;
end
```

Результат выполнения программы:

```
Таблица функции
x=0.00   y=9.654
x=0.05   y=9.865
x=0.10   y=10.307
x=0.15   y=11.017
x=0.20   y=12.069
x=0.25   y=13.580
x=0.30   y=15.750
x=0.35   y=18.916
x=0.40   y=23.688
x=0.45   y=31.230
x=0.50   y=43.976
x=0.55   y=67.649
x=0.60   y=118.059
x=0.65   y=250.835
x=0.70   y=759.486
x=0.75   y=5381.868
x=0.80   y=666746364.301
x=0.85   y=-5181.130
```

4.6. Ввод по запросу программы

Запрос на ввод создает функция ***input***().

Пример. Сохранить в файле sin*x* программу

```
x=input(' Введите аргумент: x=');
y=sin(x);
fprintf('x=%-2.1f  y=%-6.4f',x,y)
```

Выполнить программу.

***Задача* 15.** *Создать программу вычисления значения функции* $\sin(x + y)$ *для данных* x *и* y. *Данные вводить по запросу программы*.

***Задача* 16.** *Создать программу для вычисления* $\lg(x - y)$. *Вводятся любые значения* x *и* y. *Программа должна проверить, будет ли* $x - y > 0$. *Если нет, вывести соответствующее сообщение и закончить работу*.

4.7. Создание функции пользователя

Функция пользователя записывается в отдельный файл и текст функции должен начинаться с заголовка ***function***. Заголовок имеет следующий вид:

```
function [y1, y2, …] = fname(x1, x2, …)
```

Здесь *fname* – имя функции, которое задает пользователь; *y*1, *y*2, … – выходные параметры; *x*1, *x*2, … − входные параметры. Входные и/или выходные параметры могут отсутствовать.

Имя функции *fname* требуется по синтаксису заголовка, для вызова же функции используется другое имя – имя файла, в котором записана функция.

Пример. Запишем в файл *fun*1 функцию, вычисляющую значение $y = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$:

```
function y=fun(x)
      y=1/sqrt(2*pi)*exp(-x^2/2)
end
```

Для вызова функции *fun*1 можно использовать команду или создать файл со следующей программой:

```
for x=-1:0.2:1
    fun1(x);
end
```

Функция *fun*1 вызывается в цикле, при вызове значение x передается в функцию, возвращается из функции значение y, поэтому будет выведена таблица значений y.

***Задача* 17**. *Составить программу вычисления таблицы значений, не превышающих* 1000, *функции*

$$f(x) = x^4 + 2.5x^3 \sin 2.5x + (6 - 20\cos 2.5x)x^2 + 27.55\sin 2.5x + 50\cos 2.5x + N;$$

начальное значение $x = 0$, *шаг таблицы* 0.5, N *– номер варианта;* $f(x)$ *оформить как функцию. Таблица должна иметь вид*

```
x = …        y= …
x = …        y= …
```

4.8. Программирование простейшего интерфейса

Простое меню создает функция ***menu***().

Пример. Программа, создающая меню, может быть следующей:

```
m=0;
while m ~= 3
    m=menu('Выбор','Команда date','Команда keyboard','Выход');
    switch m
        case 1, today=date
        case 2, keyboard
        case 3, break
    end
end
```

Сохранить эту программу, например, в файле *demo_menu*. ***Не выполнять пока программу!***

Комментарий к программе. Команда *keyboard* вызывает командный режим работы и в этом режиме меню не действует. Чтобы возвратиться в меню, нужно выполнить команду *return*.

Выполнить программу *demo_menu*. Проверить действие различных кнопок меню.

***Задача* 18**. *Изменить следующие названия кнопок: Выбор на Доступные действия, Команда date на Вызов команды date, Команда keyboard на Вызов командного режима keyboard, Выход на Отключение меню.*

Дополнить меню пунктами вызова программ вычисления функции sin*x и* lg*x.*

***Задача* 19.** *Создать программу вычисления корней уравнения* $x^2 + \sin x - N = 0$ *с точностью* $\varepsilon = 10^{-4}$ *по алгоритму Ньютона*:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}.$$

Меню решения этой задачи должно иметь кнопку выхода в командный режим для определения начального приближения по графику левой части уравнения, кнопку вызова программы решения и кнопку проверки решения. Программа решения должна выдавать запрос на ввод начального приближения.

5. ОПЕРАЦИИ МАТЕМАТИЧЕСКОГО АНАЛИЗА В *MATLAB*

Для проведения аналитических операций, таких, как дифференцирование, интегрирование и т. д. необходимо соответствующие переменные предвари-

тельно объявить как символьные. Группу символьных переменных создает команда ***syms***, например:

```
>>  syms  s1  s2
```

5.1. Интегрирование

Интеграл вычисляет функция ***int***().

Пример. Вычислить неопределенный интеграл от функции $f(x) = x^k$:

```
>> syms x k
>> s=int(x^k,x)
s =
   x^(k+1)/(k+1)
```

***Задача* 20.** *Вычислить* $\int \frac{dx}{1+x^2}$.

Для вычисления определенного интеграла указываются пределы интегрирования, при этом бесконечный предел обозначается ***inf***.

Пример. Вычислить интеграл $\int_0^{\infty} e^{-x^2/2} dx$:

```
>> s1=int(exp(-x^2/2),x,0,inf)
s1 =
   1/2*2^(1/2)*pi^(1/2)
```

Ответ по умолчанию выводится в виде выражения − точного значения интеграла. Чтобы получить приближенное численное значение, необходимо выражение-ответ вычислить:

```
>> 1/2*2^(1/2)*pi^(1/2)
ans =
   1.2533
```

***Задача* 21.** *Найти точное и численное значения* $\int_{-3}^{6} \frac{dx}{5+x^2}$.

5.2. Дифференцирование

Дифференцирование выполняет функция ***diff***(). По умолчанию дифференцирование производится по x, при дифференцировании по другим переменным их надо явно указывать.

Пример. Найти производные первого порядка по x и по y функции $f(x, y) = \frac{x^2}{\sqrt{1+y^3}} + y\sin(x)$:

```
>> f=x^2/(1+y^3)^(1/2)+y*sin(x);
>> diff(f)
ans =
   2*x/(1+y^3)^(1/2)+y*cos(x)
>> diff(f,y)
ans =
   -3/2*x^2/(1+y^3)^(3/2)*y^2+sin(x)
```

Для вычисления производных высших порядков необходимо указать порядок производной.

Пример. Найти производную второго порядка по x функции $f(x, y) = \frac{x^2}{\sqrt{1+y^3}} + y\sin(x)$:

```
>> diff(f,x,2)
ans =
   2/(1+y^3)^(1/2) -y*sin(x)
```

***Задача* 22.** *Найти производную третьего порядка по x и четвертого порядка по y функции* $f(x,y) = \frac{x^2}{\sqrt{1+y^3}} + y\sin(x)$.

5.3. Суммирование рядов

Сумму ряда вычисляет функция ***symsum***().

Пример. Найти значение суммы $\sum_{k=1}^{\infty} \frac{1}{k^2}$:

```
>> syms k
>> symsum(k^(-2),1,inf)
ans =
   1/6*pi^2
>> 1/6*pi^2
ans =
   1.6449
```

***Задача* 23**. *Найти сумму ряда* $\sum_{n=0}^{\infty} \frac{1}{2^n} + \frac{(-1)^n}{3^n}$.

5.4. Разложение функции в ряд Тейлора

Разложение функции в ряд Тейлора выполняет функция ***taylor***(). По умолчанию вычисляется разложение до пятой степени, но можно указать другую степень.

Пример. Разложить функцию sin(x) в ряд Тейлора до пятой степени:

```
>> taylor(sin(x),x)
ans =
   x-1/6*x^3+1/120*x^5
```

Пример. Разложить функцию sin(x) в ряд Тейлора до девятой степени:

```
>> taylor(sin(x),x,10)
ans =
   x-1/6*x^3+1/120*x^5-1/5040*x^7+1/362880*x^9
```

Пример. Разложить в ряд Тейлора функцию $1-x\cos xt$:

```
>> taylor(1-x*cos(x*t),x,t)
ans =
   1-x+1/2*x^3*t^2-1/24*x^5*t^4.
```

***Задача* 24.** *Разложить в ряд Тейлора функции*: $f(x)$ = arctg x; $f(x)$ = $= \frac{x}{\sqrt{1-2x}}$; $f(x,t) = \frac{1}{\sqrt{1-2tx+x^2}}$.

5.5. Вычисление пределов

Предел вычисляет функция ***limit***().

Пример. Найти $\lim\limits_{x\to 0}\frac{\sin 5x}{x}$:

```
>> limit(sin(5*x)/x,x,0)
ans =
   5
```

Для обозначения левого и правого пределов используются слова ***'left'*** и ***'right'***.

Пример. Найти $\lim\limits_{x\to 1-0}\operatorname{arctg}\frac{1}{1-x}$ и $\lim\limits_{x\to 1+0}\operatorname{arctg}\frac{1}{1-x}$:

```
>> limit(atan(1/(1-x)),x,1,'left')
ans =
   1/2*pi
>> limit(atan(1/(1-x)),x,1,'right')
ans =
   -1/2*pi
```

Пример. Найти $\lim\limits_{x\to -\infty}\frac{\ln(1+e^x)}{x}$ и $\lim\limits_{x\to +\infty}\frac{\ln(1+e^x)}{x}$:

```
>> limit(log(1+exp(x))/x,x, -inf)
ans =
   0
>> limit(log(1+exp(x))/x,x,+inf)
ans =
   1
```

***Задача* 25.** *Найти пределы:* $\lim\limits_{x\to 0}(x+e^x)^{1/x}$; $\lim\limits_{x\to -0}\frac{1}{1+e^{1/x}}$; $\lim\limits_{x\to +0}\frac{1}{1+e^{1/x}}$.

5.6. Решение обыкновенных дифференциальных уравнений

Для решения дифференциальных уравнений в *MATLAB* применяется универсальная функция ***dsolve***(). Эта функция находит решение, если оно выражается через известные функции (элементарные или специальные). Универсальность функции состоит в том, что она находит как общее решение, так и решение задачи Коши, решает как одно уравнение, так и системы уравнений, а также решает краевые задачи для обыкновенных дифференциальных уравнений.

По умолчанию независимой переменной считается ***t***, для обозначения производных в уравнении используется символ ***D***, а комбинации ***D2***, ***D3***, … применяются для обозначения второй, третьей и последующих производных.

Пример. Найти общее решение уравнения $y'(x) = 2x(1 + y^2)$:

```
>> de='Dy=2*t*(1+y^2)'
de =
   Dy=2*t*(1+y^2)
>> S=dsolve(de)
S =
   tan(t^2+2*C1)
```

Получено общее решение $y = \mathrm{tg}(x^2 + 2C1)$, где $C1$ − произвольная постоянная. Добавим в функцию *dsolve*() начальное условие $y(0) = 0$, чтобы найти частное решение:

```
>> S=dsolve(de,'y(0)=0')
S =
   tan(t^2)
```

Пример. Решить систему $\begin{cases} y'' = y'y - y + x, \\ x' = y - 1 \end{cases}$ при $y(0) = 0,\ x(0) = 0$:

```
>> de='D2y=D1y-y+x, Dx=y-1'
de =
   D2y=D1y-y+x,Dx=y-1
>> S=dsolve(de,'y(0)=0,x(0)=0')
S =
   x: [1x1 sym]
   y: [1x1 sym]
```

Решение представлено структурой с двумя полями (по числу неизвестных функций). Вывод решения:

```
>> S.x
ans =
-1/2*exp(t)-1/2*cos(t)-1/2*sin(t)-1/2*C3*sin(t)-1/2*C3*cos(t)+1/2*C3*exp(t)+1
>> S.y
ans =
 1/2*sin(t)-1/2*cos(t)-1/2*exp(t)+1/2*C3*sin(t)-1/2*C3*cos(t)+1/2*C3*exp(t)+1
```

***Задача* 26.** *Найти общее решение уравнения* $y'' + y = 1 + e^x$ *и его частное решение при начальных условиях* $y(0) = 1,\ y'(0) = 1$.

***Задача* 27.** *Найти общее решение системы* $\begin{cases} \dfrac{dx}{dt} = -x - 2y, \\ \dfrac{dy}{dt} = 3x + 4y, \end{cases}$ *и ее частное решение при начальных условиях* $x(0) = 0,\ y(0) = 1$.

6. АЛГОРИТМЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

6.1. Свойства алгоритмов вычислительной математики

Результатом решения задачи в вычислительной математике должны быть числа. Например, в теоретической математике ответом может быть $x = \sqrt{2}$. В вычислительной математике это не ответ, потому что $\sqrt{2}$ − не число, ответом может быть $x = 1.41$ или $x = 1.414$, или $x = 1.4142$, в зависимости от требуемой точности представления результата извлечения квадратного корня из 2.

Вычислительная математика разрабатывает методы доведения до числового результата решений основных задач математического анализа, алгебры и геометрии. Эти методы называются численными методами.

Численные методы имеют свой язык записи. Чтобы численный метод давал в результате число, он должен быть представлен в виде чисел и арифметических операций над числами. Однако задачи формулируются обычно на математическом языке (языке уравнений, функций, производных, интегралов и т. п.). Поэтому разработка численного метода состоит в том, чтобы решение исходной задачи представить в виде арифметических операций над числами.

Общими принципами при разработке численных методов являются:

1) решение исходной задачи, сформулированной на математическом языке, заменяется другой задачей – вычислительным алгоритмом;

2) вычислительный алгоритм содержит некоторый параметр *k*, которого нет в исходной задаче;

3) выбором параметра *k* можно добиться любой близости решения второй задачи к решению первой;

4) неточная реализация алгоритма, вызванная округлениями, не меняет существенно свойств алгоритма.

Пример. Найти численное значение выражения $\sqrt{2}$ с точностью $\varepsilon = 10^{-6}$.

Поступим следующим образом. В качестве алгоритма численного решения этой задачи возьмем формулу $x^{(k+1)} = \frac{1}{2}\left(x^{(k)} + \frac{2}{x^{(k)}}\right)$ (вывод формулы будет

дальше). Возьмем какое-либо начальное приближение $x^{(0)}$ (например $x^{(0)} = 1$) и будем последовательно вычислять значения $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, … по данному алгоритму, последовательно получим:

$x^{(1)} = 1.5$; $x^{(2)} = 1.416667$; $x^{(3)} = 1.414216$; $x^{(4)} = 1.414216$; $x^{(5)} = 1.414216$; …

Начиная с $x^{(3)}$, шесть цифр в дробной части не меняются, поэтому $\sqrt{2}$ = = 1.414216 с точностью 10^{-6}.

***Задача* 28.** *Вычислить* $\sqrt{a}$, *где* $a = N + 2$, *если корень из числа* $N + 2$ *извлекается точно, значение* $N + 2$ *увеличить на* 1. *Использовать алгоритм* $x^{(k+1)} = \frac{1}{2}\left(x^{(k)} + \frac{a}{x^{(k)}}\right)$.

6.2. Алгоритмы и функции *MATLAB* для решения уравнения *f*(*x*) = 0

Алгоритм Ньютона. Построим алгоритм решения уравнения $f(x) = 0$. По графику левой части уравнения определим точку $x^{(k)}$, близкую к точке пересечения графика с осью x. Разложим функцию $f(x)$ в ряд Тейлора в окрестности точки $x = x^{(k)}$:

$$f(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{f''(x^{(k)})}{2!}(x - x^{(k)})^2 + \ldots$$

Отбросим справа все слагаемые, содержащие производные выше первого порядка, тогда получим приближенное равенство

$$f(x) \approx f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}).$$

Обозначим $x^{(k+1)}$ корень уравнения $f(x) = 0$ и подставим его в приближенное равенство, получим $f(x^{(k+1)}) \approx f(x^{(k)}) + f'(x^{(k)})(x^{(k+1)} - x^{(k)})$. Если $x^{(k+1)}$ найти из уравнения $f(x^{(k)}) + f'(x^{(k)})(x^{(k+1)} - x^{(k)}) = 0$, то оно будет приближенным решением уравнения $f(x) = 0$. Отсюда получаем алгоритм

$$x^{(k+1)} = x^{(k)} + \frac{f(x^{(k)})}{f'(x^{(k)})}.$$

Этот алгоритм вычисления корня уравнения $f(x) = 0$ называется алгоритмом Ньютона. Проводя вычисления для $k = 0, 1, 2, \ldots$ получим последовательность чисел, которая сходится к точному значению корня при $k \to \infty$. Практически уточнения корня производят, пока выполняется неравенство $| x^{(k+1)} - x^{(k)} | > \varepsilon$, где ε − заданная точность.

***Задача* 29.** *Вычислить с точностью* $\varepsilon = 10^{-6}$ *корни уравнения* $5x^3 + Nx^2 - 15x - 6 = 0$ *по алгоритму Ньютона. Проверить корни.*

***Задача* 30.** *Используя алгоритм Ньютона, построить алгоритм вычисления* $\sqrt[n]{a}$ *. Вычислить* $\sqrt[5]{2}$ *с точностью* $\varepsilon = 10^{-6}$.

Алгоритм секущих. Алгоритм секущих можно получить из алгоритма Ньютона при замене производной разностным отношением. Точное значение производной в точке $x^{(k)}$ определяется равенством

$$f'(x^{(k)}) = \lim_{x^{(k-1)} \to x^{(k)}} \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}.$$

Если исключить операцию вычисления предела, получим приближенное значение производной в точке $x^{(k)}$:

$$f'(x^{(k)}) \approx \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}.$$

Заменив в алгоритме Ньютона производную ее приближенным значением, получим алгоритм секущих:

$$x^{(k+1)} = \frac{f(x^{(k)})x^{(k-1)} - f(x^{(k-1)})x^{(k)}}{f(x^{(k)}) - f(x^{(k-1)})}.$$

Приближения продолжаются, пока выполняется условие $|x^{(k+1)} - x^{(k)}| > \varepsilon$.

В алгоритме секущих используются два предыдущих приближения $x^{(k)}$ и $x^{(k-1)}$, поэтому при взятом начальном приближении $x^{(0)}$ необходимо вычислить следующее приближение $x^{(1)}$ по алгоритму Ньютона с приближенной заменой производной по формуле $f'(x^{(0)}) \approx \frac{f(x^{(0)}+\varepsilon) - f(x^{(0)})}{\varepsilon}$, тогда $x^{(1)} = \frac{f(x^{(0)})\varepsilon}{f(x^{(0)}+\varepsilon) - f(x^{(0)})}$.

***Задача* 31.** *Вычислить с точностью* $\varepsilon = 10^{-6}$ *корни уравнения* $5x^3 + Nx^2 - 15x - 6 = 0$ *по алгоритму секущих.*

Алгоритм деления отрезка пополам. Пусть непрерывная функция $f(x)$ принимает на концах отрезка $[a, b]$ значения разных знаков, тогда корень уравнения $f(x) = 0$ содержится в интервале (a, b). Разделим интервал на две половины и дальше будем рассматривать ту половину, на концах которой функция принимает значения разных знаков. Этот новый отрезок делим на два равных отрезка и т. д.

Процесс деления продолжается до тех пор, пока длина очередного отрезка не станет меньше требуемой величины погрешности.

Запишем алгоритм половинного деления по пунктам:

0) определить значения a, b, ε;
1) вычислить $x = (a + b)/2$; $f(x)$;
2) если $f(x) = 0$, перейти к п. 5;
3) если $f(x) \cdot f(a) < 0$ положить $b = x$, иначе $a = x$;
4) если $|b - a| > \varepsilon$ перейти к п. 1;
5) вывести значение x; конец.

Задача 32. *Вычислить с точностью* $\varepsilon = 10^{-6}$ *меньший корень уравнения* $5x^3 + Nx^2 - 15x - 6 = 0$ *по алгоритму половинного деления.*

Функции MATLAB для решения уравнения f(x) = 0

Основные функции:

1) ***fzero***('*f*(*x*)' $x^{(0)}$) − функция возвращает значение, близкое к корню или *NaN,* если такая точка не найдена;

2) ***fzero***('*f*(*x*)' [*a*, *b*]), где [*a*, *b*] − интервал поиска корня такой, что знак *f*(*a*) отличается от знака *f*(*b*); если это условие не выполняется, выдается сообщение об ошибке;

3) ***fzero***('*f*(*x*)' $x^{(0)}$, *eps*).

Все функции *fzero*() находят корень, если график левой части уравнения *f*(*x*) = 0 пересекает ось *x*. Если график касается оси *x*, то точка касания также является корнем, но такой корень функции *fzero*() определить не могут. Кроме того, эти функции могут вычислить только один корень, для вычисления других имеющихся корней приходится повторно использовать функцию *fzero*().

Имеются функции ***fsolve***() и ***solve***(), предназначенные для решения систем нелинейных уравнений, однако их можно использовать и для решения одного уравнения. С помощью этих функций можно найти все корни на заданном промежутке, а также корни, являющиеся точками касания графика оси *x*.

Рассмотрим использование этих функций на примере решения уравнения 0.25*x* + sin*x* – 1 = 0. Построим график левой части

```
>> ezplot('0.25*x+sin(x) -1', [-5  15]), grid
```

График (рис. 6.1) показывает, что уравнение имеет три корня на промежутке [0; 10]. Вычислим эти корни с помощью функции *fsolve*():

```
>> fsolve('0.25*x+sin(x) -1',0:10)
ans =
  Columns 1 through 8
    0.8905  0.8905  2.8500  2.8500  2.8500  5.8128  5.8128  5.8128
  Columns 9 through 11
    5.8128  10.7429  10.7429
```

Согласуя вывод с графиком, получим $x_1 = 0.8905$, $x_2 = 2.85$, $x_3 = 5.8128$.

Используя функцию *solve*(), вычисляем наименьший корень:

```
>> solve('0.25*x+sin(x) -1')
ans =
   .89048708074438001001103173059554
```

Задача 33. *Найти корни уравнения* $x^3 \cos x - 2x \sin x + N = 0$ *с помощью функции MATLAB.*

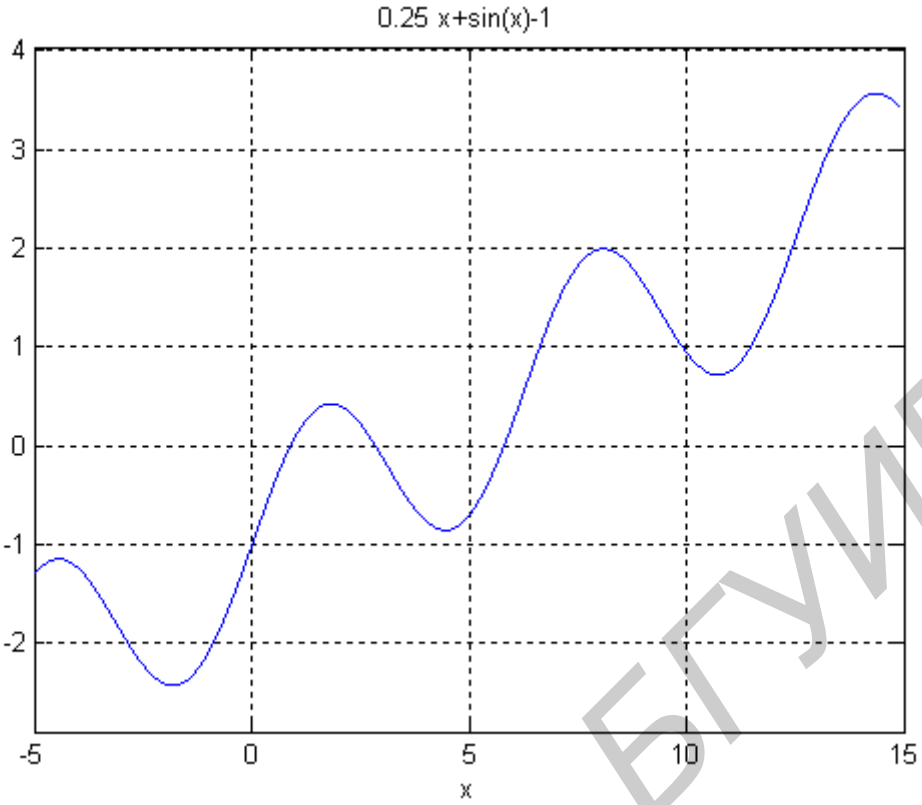


Рис. 6.1

6.3. Решение систем нелинейных алгебраических уравнений

Системы нелинейных уравнений $\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0, \\ \ldots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{cases}$ решают практически только итерационными методами: пусть известно некоторое приближение $x^{(k)}$ к корню x^*, тогда следующее приближение находят по формуле

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)},$$

где $\Delta x^{(k)}$ – решение системы линейных уравнений

$$\sum_{i=1}^{n} \frac{\partial f_j(x^{(k)})}{\partial x_i} \Delta x_i^{(k)} = -f_j(x^{(k)}), \quad j = 1, 2, \ldots, n.$$

Данный метод решения системы нелинейных уравнений называется методом Ньютона. Итерации методом Ньютона продолжают до тех пор, пока выполняется $|\Delta x^{(k)}| > \varepsilon$, где ε – заданная точность решения.

Нулевое приближение $x^{(0)}$ в случае двух переменных можно найти графически: построить на плоскости кривые $f_1(x_1, x_2) = 0$ и $f_2(x_1, x_2) = 0$ и найти точки их пересечения. Для трех и более переменных удовлетворительных способов подбора нулевых приближений нет.

***Задача* 34 .** *Решить методом Ньютона с точностью* $\varepsilon = 10^{-5}$ *систему*

$$\begin{cases} (1 + N/10)x^3 - y^2 - 1 = 0, \\ xy^3 - y - 4 = 0. \end{cases}$$

Выполнить проверку решения.

Меню программы решения этой задачи должно иметь кнопку для определения начального приближения графическим методом, кнопку вызова программы решения и кнопку проверки решения. Программа решения должна выдавать запрос на ввод начального приближения.

В *MATLAB* решение системы нелинейных уравнений можно найти с помощью функции ***solve***().

Пример. Используем функцию *solve*() для решения системы $\begin{cases} x = y^3, \\ x + y = 1 \end{cases}$

```
>>s= solve('x=y^3','x+y=1')
ans =
    x: [3x1 sym]
    y: [3x1 sym]
```

Функция *solve* находит аналитическое решение, и для хранения результата создается структура с полями, соответствующими именам неизвестных. В этом примере имя структуры *s* , в поле *x* хранятся значения неизвестной *x*, в поле *y* − значения неизвестной *y*. Эти значения можно вывести:

```
>> s.x
ans =
[ 1/6*(108+12*93^(1/2))^(1/3)+2/(108+12*93^(1/2))^(1/3)+1]
[ 1/12*(108+12*93^(1/2))^(1/3)-1/(108+12*93^(1/2))^(1/3)-
1/2*i*3^(1/2)*(1/6*(108+12*93^(1/2))^(1/3)+2/(108+12*93^(1/2))^(1/3))+1]
[ 1/12*(108+12*93^(1/2))^(1/3)-
1/(108+12*93^(1/2))^(1/3)+1/2*i*3^(1/2)*(1/6*(108+12*93^(1/2))^(1/3)+2/(1
08+12*93^(1/2))^(1/3))+1]
```

Можно вывести и отдельные значения полей, например, выведем первое значение поля *x*:

```
>> s.x(1)
ans =
  -1/6*(108+12*93^(1/2))^(1/3)+2/(108+12*93^(1/2))^(1/3)+1
```

Аналитические выражения корней громоздки. Чтобы преобразовать результат вычисления корней в числовой вид, используют функцию ***subexpr***():

```
>> [r, q]=subexpr(s. x); r
r =
   0.3177
   1.3412 - 1.1615i
   1.3412 + 1.1615i
>> [r, q]=subexpr(s. y); r
r =
   0.6823
  -0.3412 + 1.1615i
  -0.3412 - 1.1615i
```

Массив ***r*** содержит корни, а массив ***q*** вспомогательный, его можно не выводить.

Задача* 35.** *Решить с помощью функции* ***solve() *систему уравнений* $\begin{cases}(1+N/10)x^3 - y^2 - 1 = 0,\\ xy^3 - y - 4 = 0.\end{cases}$ *Решение проверить.*

***Задача* 36.** *Решить систему* $\begin{cases}x^2 + y^2 + z^2 = N,\\ 2x^2 + y^2 - 4z = 0,\\ 3x^2 - 4y + z^2 = 0.\end{cases}$

6.4. Численное решение обыкновенных дифференциальных уравнений

Рассмотрим задачу Коши для обыкновенного дифференциального уравнения первого порядка $y' = f(x, y)$, $y(x_0) = y_0$. Требуется найти функцию $y = y(x)$, которая удовлетворяет уравнению на интервале $[x_0, X]$ и начальному условию в точке x_0.

Точные методы решения дифференциальных уравнений известны только для некоторых классов уравнений. Из численных методов одним из наиболее употребительных является метод Рунге – Кутта четвертого порядка: отрезок $[x_0, X]$ разбивают на n частей точками $x_i = x_0 + ih$, $h = \dfrac{X - x_0}{n}$ и на каждом i-м шаге вычисляют коэффициенты

$$k_1 = f(x_i, y_i),$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right),$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right),$$

$$k_4 = f(x_i + h, y_i + hk_3).$$

Решение y_{i+1} находят по формуле $y_{i+1} = y_i + \dfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$.

Погрешность метода Рунге − Кутта четвертого порядка равна $O(h^4)$ на всем промежутке интегрирования дифференциального уравнения.

***Задача* 37.** *Найти по алгоритму Рунге − Кутта решение задачи Коши* $y'(x) = 2x(1 + y^2)$, $y(0) = 0$ *на промежутке* $[0; 1]$ *с шагом* $h = 0.2$.

Ответ:

x	***y*(*x*)**
0	0
0.2	0.04
0.4	0.161
0.6	0.376
0.8	0.745
1.0	1.557

Эта система имеет точное решение $y = \mathrm{tg}(x^2)$.

Для численного решения одного уравнения или системы дифференциальных уравнений методом Рунге − Кутта четвертого порядка в *MATLAB* имеется функция ***ode*45**(), формат вызова которой имеет вид

```
>> [T,Y]=ode45('rhs',[ Tspan],[y0],[options])
```

Результаты решения задачи Коши размещаются в матрице *Y*, вектор *T* содержит значения независимой переменной *t, rhs* − это имя файла, в который записывается правая часть уравнения. Этот файл необходимо подготовить отдельно. *Tspan* − массив, определяющий интервал интегрирования; *y0* − вектор начальных условий, параметр *options* можно не использовать, но пустые скобки нужно оставить.

Пример. Используя функцию ***ode*45**(), решим задачу Коши $y'(x) = 2x(1+y^2)$, $y(0) = 0$ на промежутке [0; 1]. Создадим файл с именем ***rhs*** следующего содержания:

```
function f=rhs(t,y)
f=2*t*(1+y.^2);
```

Выполняем команду

```
>> [T,Y]=ode45('rhs',[0 1],[0 0],[])
```

Построим на одном чертеже (рис. 6.2) графики точного (сплошной линией) и приближенного (кружками) решений дифференциального уравнения

```
>> z=tan(T.^2);plot(T,Y,T,z,'o')
```

***Задача* 38.** *Решить задачу Коши* $y'(t) = \frac{y}{t} + t^2$, $y(0.5) = 0.5$ *на промежутке* [0.5; 2]. *Построить на одном чертеже графики точного и приближенного решений дифференциального уравнения. Точное решение* $y = \frac{t^3}{2} + 0.875t$.

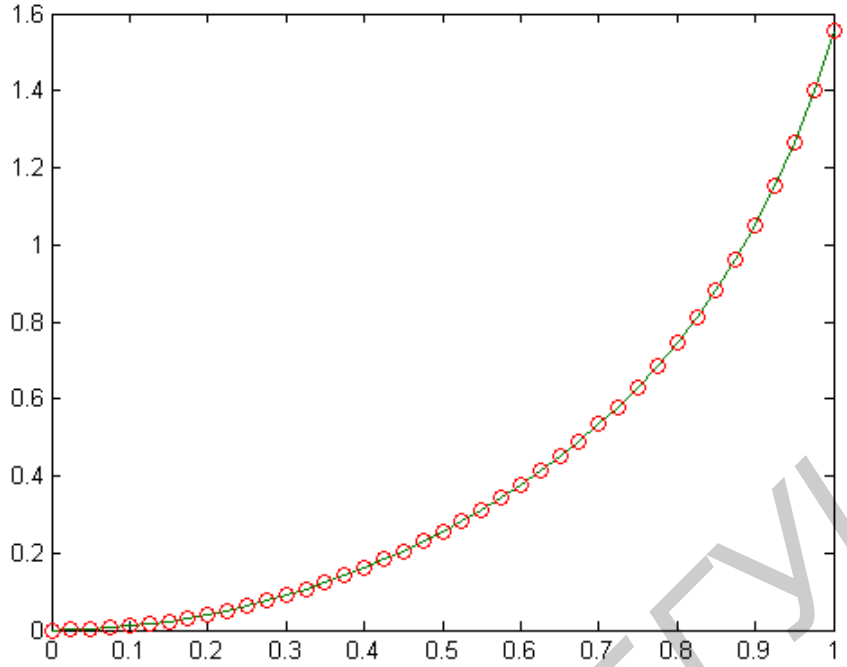


Рис. 6.2

6.5. Аппроксимация функций

Рассмотрим аппроксимацию дискретных функций, т. е. функций $y = f(x)$, задаваемых таблицей:

x_0	$y_0 = f(x_0)$
x_1	$y_1 = f(x_1)$
. . .	. . .
x_n	$y_n = f(x_n)$

Для вычисления $f(x)$ в произвольной точке промежутка $[x_0, x_n]$ строят непрерывную функцию (обычно полином), проходящую через заданные точки. Такой полином называют интерполяционным, а значения функции, вычисленные по интерполяционному полиному, называют интерполяционными.

В *MATLAB* для вычисления коэффициентов аппроксимирующего полинома степени n таблицы данных X и Y используется функция ***polyfit***(X, Y, n). При выводе коэффициенты располагаются в следующем порядке: коэффициент при старшей степени полинома выводится первым и далее по убыванию степеней.

Пример. Функция $y = f(t)$ задана таблицей:

t	0	7	12	17	22	27	32	37
y	100	87.3	72.9	63.2	54.7	47.5	41.4	36.3

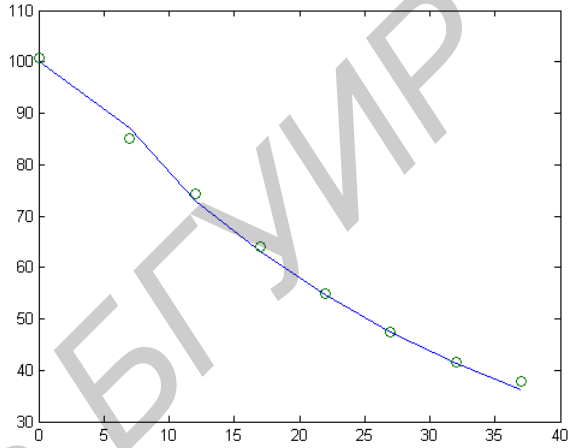
Построить полином третьей степени, аппроксимирующий данную функцию:

```
>> t=[0 7 12 17 22 27 32 37];
>> y=[100 87.3 72.9 63.2 54.7 47.5 41.4 36.3];
>> p=polyfit(t,y,3)
p =
    0.0005   -0.0044   -2.2192  100.6563
```

Ответ: $y(t) = 0.000t^3 - 0.0044t^2 - 2.2192t + 100.6563$

Построим на одном чертеже графики исходных данных и аппроксимирующего полинома

```
>> Y=0.0005*t.^3-0.0044*t.^2
 -2.2192*t+100.6563;
  plot(t,y,t, Y,'o')
```



***Задача* 39.** *Функция* $y = f(t)$ *задана таблицей:*

t	0	7	12	17	22	27	32	37
y	100	87.3	72.9	63.2	54.7	47.5	41.4	36.3

Построить полином пятой степени, аппроксимирующий данную функцию, и графики исходных данных и полинома.

В *MATLAB* для одномерного интерполирования используется функция ***interp*1**(*X, Y, Xi, 'Method'*), где *X*, *Y* − заданные массивы значений аргумента и функции; *Xi* − массив значений аргумента, при которых требуется вычислить интерполяционные значения функции. Параметр *Method* задает метод интерполирования: по ближайшей точке (*nearest*), линейная интерполяция (*linear*) (действует по умолчанию), кубическая (*cubic*) или интерполяция кубическим сплайном (*spline*). Для всех методов величины элементов массива *X* должны меняться монотонно. Для точек *Xi*, лежащих вне интервала, определенного *X*, присваиваются константы *NaN*.

***Задача* 40.** *Функция* $f(x)$ *задана следующей таблицей*:

x	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
***f*(*x*)**	0.5652	0.6375	0.7147	0.7973	0.8861	0.9817	1.0848	1.1964	1.3172	1.4482	1.5906

Найти значение $f(x)$ *для* $x = 1.03N$, *используя различные методы компьютерного интерполирования.*

Сплайн (от *spline* − сращивание) отличается от кубического интерполяционного полинома тем, что он имеет одну формулу для всего промежутка $[x_0, x_n]$, а кубический полином для каждого частичного промежутка имеет другие коэффициенты.

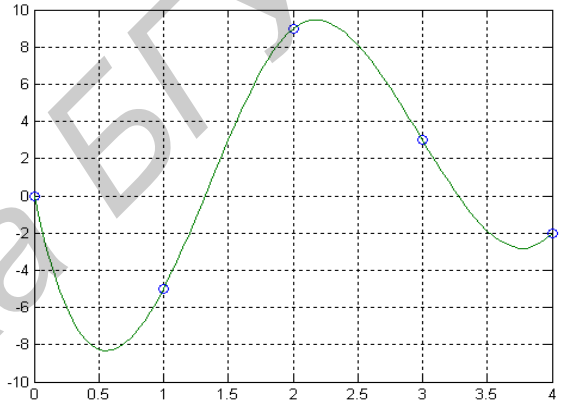
В *MATLAB* значения кубического сплайна для значений *Xi* по таблице (*X*, *Y*) выведет функция *interp*1(*X, Y, Xi, 'spline'*). Но имеется и специальная функция для построения сплайна ***spline***(*X, Y, Xi*).

Пример. Вычислить *f*(1.23), используя кубический сплайн и следующую табличную функцию:

x	0	1	2	3	4
y	0	–5	9	3	–2

Решение:

```
>> X=[0 1 2 3 4];Y=[0 –5 9 3 –2];
>> Yi=spline(X,Y,1.23)
Yi =
   –1.502
```



Построим график кубического сплайна и табличные точки. Для построения графика необходимо сгустить сетку *Xi = 0:0.05:4*, используя команду:

```
>> X=[0 1 2 3 4];Y=[0 –5 9 3 –2];
   Xi=0:0.05:4;
   Yi=spline(X,Y,Xi);
>> plot(X, Y, 'o', Xi, Yi, '–') , grid
```

***Задача* 41.** *Вычислить f* (2.55), *используя кубический сплайн и функцию interp*1(*X, Y, Xi, ='spline'*), *по предыдущей таблице*.

Очень эффективным способом аппроксимации данных является метод наименьших квадратов, который состоит в следующем.

Пусть при проведении некоторого опыта наблюдаются две величины *X* и *Y*, тогда *n* независимых повторений опыта дадут *n* пар наблюдавшихся значений (x_1, y_1), (x_2, y_2), …, (x_n, y_n). Если точки группируются вблизи некоторой линии, то можно подобрать аналитическую форму зависимости, соответствующую графику этой линии.

Пусть выбрана квадратичная зависимость (парабола) $y = a + bx + cx^2$. Для оценки параметров *a*, *b* и *c* используем метод наименьших квадратов: составим сумму квадратов отклонений ординат точек, расположенных на параболе, от ординат опытных точек $S = \sum_{i=1}^{n} (a + bx_i + cx^2 - y_i)^2$. Параметры *a*, *b* и *c* определяют так, чтобы сумма квадратов отклонений была минимальной. Условие минимума *S* относительно *a*, *b* и *c* дает систему уравнений:

$$\begin{cases} \dfrac{\partial S}{\partial a} = \dfrac{\partial}{\partial a}\sum\limits_{i=1}^{n}(a+bx_i+cx_i^2-y_i)^2 = 0, \\ \dfrac{\partial S}{\partial b} = \dfrac{\partial}{\partial b}\sum\limits_{i=1}^{n}(a+bx_i+cx_i^2-y_i)^2 = 0, \\ \dfrac{\partial S}{\partial c} = \dfrac{\partial}{\partial c}\sum\limits_{i=1}^{n}(a+bx_i+cx_i^2-y_i)^2 = 0 \end{cases} \quad \text{или} \quad \begin{cases} na+\left(\sum\limits_{i=1}^{n}x_i\right)b+\left(\sum\limits_{i=1}^{n}x_i^2\right)c=\sum\limits_{i=1}^{n}y_i, \\ \left(\sum\limits_{i=1}^{n}x_i\right)a+\left(\sum\limits_{i=1}^{n}x_i^2\right)b+\left(\sum\limits_{i=1}^{n}x_i^3\right)c=\sum\limits_{i=1}^{n}x_iy_i, \\ \left(\sum\limits_{i=1}^{n}x_i^2\right)a+\left(\sum\limits_{i=1}^{n}x_i^3\right)b+\left(\sum\limits_{i=1}^{n}x_i^4\right)c=\sum\limits_{i=1}^{n}x_i^2y_i. \end{cases}$$

Решив эту систему, получим коэффициенты a, b и c, и задача построения аппроксимирующего полинома решена.

Для суммирования элементов матрицы A имеется функция ***sum*** (A), которая вычисляет сумму элементов матрицы по столбцам. Если A − вектор, то *sum*(A) вычисляет сумму элементов вектора.

Задача* 42**. *При исследовании некоторой химической реакции через каждые* 5 *мин определялось количество* ***y *вещества, оставшееся в системе. Результаты измерений приведены в таблице, где* ***t*** *− время после начала реакции в мин;* ***y*** *− количество вещества в процентах*:

t	0	7	12	17	22	27	32	37
y	100	87.3	72.9	63.2	54.7	47.5	41.4	36.3

Полагая, что t и y связаны зависимостью $y = a + bt + ct^2$, *найти коэффициенты a, b и c методом наименьших квадратов.*

Определить, какой процент вещества остается в системе по истечении 25 *мин после начала реакции.*

Построить на одном чертеже исходные точки и график аппроксимирующей зависимости.

Ответ: $y = 101.4 - 2.58t + 0.022t^2$; $y(25) = 50.6\%$.

Решение этой задачи можно представить одним из трех уровней сложности.

Первый уровень. *Решение выполнить в командном режиме.*

Второй уровень. *Составить программу решения без меню.*

Третий уровень. *Создать меню решения, в котором должны быть следующие кнопки: ввод векторов t и y, вывод аналитического выражения зависимости, ввод значения аргумента и вывод значения функции, вывод графиков.*

6.6. Численное дифференцирование и интегрирование

Пусть функция $y = f(x)$ задана в равноотстоящих точках x_i (i = 0, 1, 2, …, n) отрезка $[a, b]$ значениями $y_i = f(x_i)$.

Составим таблицу разностей вида:

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
x_0	y_0	Δy_0	$\Delta^2 y_0$	$\Delta^3 y_0$	$\Delta^4 y_0$	$\Delta^5 0$
x_1	y_1	Δy_1	$\Delta^2 y_1$	$\Delta^3 y_1$	$\Delta^4 y_1$	
x_2	y_2	Δy_2	$\Delta^2 y_2$	$\Delta^3 y_2$		
x_3	y_3	Δy_3	$\Delta^2 y_3$			
x_4	y_4	Δy_4				
x_5	y_5					

Разности вычисляют и записывают, начиная с низа соответствующего столбца. Верхний индекс i выражения $\Delta^i y$ называется порядком разности. Первая и вторая производные функции $y = f(x)$ в основных табличных точках вычисляются по формулам

$$y'(x_0) = \frac{1}{h}\left(\Delta y_0 - \frac{\Delta^2 y_0}{2} + \frac{\Delta^3 y_0}{3} - \frac{\Delta^4 y_0}{4} + \frac{\Delta^5 y_0}{5} - \dots \right),$$

$$y''(x_0) = \frac{1}{h^2}\left(\Delta^2 y_0 - \Delta^3 y_0 + \frac{11}{12}\Delta^4 y_0 - \frac{5}{6}\Delta^5 y_0 + \dots \right),$$

где $h = x_{k+1} - x_k$;

x_0 – начальная точка (в качестве начальной точки можно взять любое табличное значение).

При вычислении производных используют ту строку разностей, в которой находится выбранное начальное приближение.

***Задача* 43.** *Путь $y = f(t)$, пройденный прямолинейно движущейся точкой за время t, задан следующей таблицей*:

Время t_i, с	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
Путь $y(t_i)$, см	0.000	1.519	6.031	13.397	23.396	35.721	50.000	65.798	82.635	100.000

Используя конечные разности до пятого порядка включительно, приближенно найти скорость $v = \frac{dy}{dt}$ и ускорение $w = \frac{d^2 y}{dt^2}$ точки для момента времени $t = 0{,}03$.

Ответ: $v = 873.2,\ w = 26250$.

Пусть для данной функции $y = f(x)$ требуется найти $\int_a^b y dx$. Вычислить интеграл по формуле Ньютона – Лейбница $\int_a^b f(x)dx = F(b) - F(a)$ во многих случаях затруднительно или даже практически невозможно, тогда прибегают к численным методам интегрирования.

Одним из методов численного интегрирования является метод парабол (метод Симпсона): возьмем четное число n и на промежутке интегрирования $[a, b]$ выбираем равноотстоящие точки $a = x_0 < x_1 < \ldots < x_n = b$, вычислим в этих точках значения подынтегральной функции $y_i = f(x_i)$, тогда приближенное значение интеграла можно найти по формуле парабол

$$\int_a^b y dx \approx \frac{h}{3}\left[(y_0 + y_n) + 4(y_1 + y_3 + \ldots y_{n-1}) + 2(y_2 + y_4 + \ldots y_{n-2})\right], \quad h = \frac{b-a}{n}.$$

Для оценки погрешности формулы парабол на практике часто начинают вычисление с некоторым четным значением n, затем проводят вычисление с удвоенным значением n. Считают, что совпадающие десятичные знаки обоих результатов принадлежат точному значению интеграла, и за приближенное значение интеграла берут любое из найденных значений. Если точность недостаточна, снова удваивают число шагов.

***Задача* 44.** *Вычислить значение интеграла* $\int_0^1 \frac{\sin t}{t} dt$ *по формуле парабол. Оценить погрешность результата.*

Ответ: точное значение $\int_0^1 \frac{\sin t}{t} dt = Si(1)$, *где* $Si(x)$ − *специальная функция* (*интегральный синус*); $Si(1) \approx 0.9460830704$.

ЛИТЕРАТУРА

1. Дьяконов, В. П. *MATLAB* 6 : учеб. курс / В. П. Дьяконов. – СПб. : Питер, 2002.

2. Дьяконов, В. П. *MATLAB* 7.* /R2006 /R2007 : самоучитель / В. П. Дьяконов. – М. : ДМК, 2008.

3. Соболь, Б. В. Практикум по вычислительной математике / Б. В. Соболь, Б. Ч. Месхи, И. М. Пешхоев. − Ростов н/Д : Феникс, 2008.

4. Калиткин, Н. Н. Численные методы / Н. Н. Калиткин. – М. : Наука, 1978.

5. Бахвалов, Н. С. Численные методы / Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков. – М. : Наука, 1987.

6. Копченова, Н. В. Вычислительная математика в примерах и задачах / Н. В. Копченова, И. А. Марон. – М : Наука, 1972.

Св. план 2011, поз. 41

Учебное издание

Самуйлов Александр Захарович
Кривоносова Татьяна Михайловна

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА С *MATLAB*

Методическое пособие
по дисциплине «Вычислительная математика»
для студентов специальностей
1-39 02 01 «Моделирование и компьютерное проектирование радиоэлектронных средств», 1-39 01 01 «Радиотехника»,
1-39 01 03 «Радиоинформатика»,
1-38 02 03 «Техническое обеспечение безопасности»
всех форм обучения

Редактор Е. Н. Батурчик
Корректор А. В. Тюхай
Компьютерная верстка В. М. Задоля

Подписано в печать	Формат 60x84 1/16.	Бумага офсетная.
Гарнитура «Таймс».	Отпечатано на ризографе.	Усл. печ. л.
Уч.-изд. л. 3,0.	Тираж 100 экз.	Заказ 798.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02370/0494175 от 03.07.2009.
220013, Минск, П. Бровки, 6