Лабораторная работа № 1

ЦЕЛЬ РАБОТЫ: Ознакомиться с вводом вещественных и комплексных чисел различного формата системе MatLAB

СОДЕРЖАНИЕ РАБОТЫ:

1. Форматы вещественных чисел
2. Основные математические операции, выполняемые системой MatLAB
3. Особенности присвоения имен переменным
4. Промежуточные переменные системы MatLAB
5. Ввод комплексных чисел
6. Основные функции комплексных чисел:

- мнимая часть комплексного числа
- вещественная часть комплексного числа
- аргумент комплексного числа
- модуль комплексного числа

7. Комплексно сопряженные числа
8. Действия над комплексными числами
9. Выполнить практическую часть и оформить отчет о проделанной работе (введенные выражения, полученные на экране результаты).
10. Ответить на контрольные вопросы.

ПРАКТИЧЕСКАЯ ЧАСТЬ

1) Ввести в окне управления математическое выражение:

$$y=(N^2+45*N)*4.5/N \quad (1),$$

где N - номер вашей машины и получить результат.

2) Получить тот же результат с использованием промежуточных переменных **ans** и **dysp.**

3) Ввести в окне управления математическое выражение 1 без вывода промежуточных значений на экран.

2) Ввести в окне управления комплексное число вида:

$$x= 5 * N + 4.5*N*i \quad (2)$$

где N - номер вашей машины.

5) Умножить **xl** на число (N +2) и присвоить это значение переменной **x2**

5) Выполнить следующие математические операции с комплексными числами **xl** и **x2**: сложение, вычитание, умножение, деление слева направо и справа налево.

6) Найти функции комплексных чисел xl и x2: мнимую и действительную части, модуль, аргумент, комплексно сопряженное число.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие форматы чисел используются системой MatLAB
2. Какие промежуточные переменные используются системой MatLAB и для чего?
3. Как избежать вывода на экран всех промежуточных результатов?
4. Какие математические операции предусмотрены в системе MatLAB для вещественных чисел?
5. Как вводятся комплексные числа в системе MatLAB
6. Какие специальные функции предусмотрены в системе MatLAB для комплексных чисел
7. Какие математические операции предусмотрены в системе MatLAB для комплексных чисел?

Лабораторная работа № 2

ЦЕЛЬ РАБОТЫ: Ознакомиться с вводом вещественных и комплексных матриц и векторов различного формата системе MatLAB и математическими операциями над ними.

СОДЕРЖАНИЕ РАБОТЫ:

1. Создание векторов и матриц вещественных чисел
2. Создание векторов и матриц комплексных чисел
3. Операции по перемещению столбцов и строк
4. Создание специальных векторов и матриц
5. Основные математические операции, выполняемые системой MatLAB над матрицами
6. Получение графиков векторов
7. Практическая часть и оформление отчета о проделанной работе (введенные выражения, полученные на экране результаты).
8. Контрольные вопросы.

ПРАКТИЧЕСКАЯ ЧАСТЬ

1) Ввести в окне управления матрицу A размерностью 5x5 состоящую из прямой последовательности вещественных чисел начиная с N - номера вашей машины (например 3, 4, 5, 6, ... 28)
2) Получить из матрицы A матрицу, строки которой переставлены относительно горизонтальной оси.
3) Получить из матрицы A матрицу, столбцы которой переставлены относительно вертикальной оси.
4) Ввести в окне управления единичный вектор - строку C, состоящий из (N+2) элементов (N - номер вашей машины)
5) Преобразовать вектор - строку C в вектор - столбец B
6) Создать из вектора, состоящего из (N+4) элементов, квадратную матрицу F с «единицами» по главной диагонали, «двойками»- по диагонали ниже главной и «тройками» по диагонали выше главной. Остальные элементы нули.
7) Создать из матрицы F квадратную матрицу D меньшего размера, исключив первые и последние столбцы и строки.
8) Выполнить сложение и вычитание матрицы F и обратной ей матрицы F"[1]
9) Выполнить умножение и деление слева направо матрицы D на вектор-столбец B.
10) Создать вектор - строку из комплексных чисел: мнимая часть - вектор C, действительная часть - вектор C, умноженный на N - номер вашей машины.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как вводятся вектора и матрицы в системе MatLAB?
2. Какие операции по преобразованию матриц позволяет выполнять система MatLAB?
3. Какие математические операции предусмотрены в системе MatLAB для матриц?
4. Как вводятся комплексные матрицы в системе MatLAB?
5. Какие специальные функции предусмотрены в системе MatLAB при создании матриц?

ПРОСТЕЙШИЕ ОПЕРАЦИИ С ВЕКТОРАМИ И МАТРИЦАМИ

MatLAB является системой, которая специально предназначена для осуществления сложных вычислений с векторами, матрицами и полиномами.

Под вектором в MatLAB понимается одномерный массив чисел, а под матрицей — двумерный массив. При этом по умолчанию предполагается, что любая заданная переменная является вектором или матрицей. Например, отдельное заданное число программа воспринимает как матрицу размером (1*1), а вектор-строку с N элементами — как матрицу размером (1*N).

1. Ввод векторов и матриц

Исходные значения векторов можно задавать с клавиатуры путем поэлементного ввода. Для этого в строке следует вначале указать имя вектора, потом поставить знак присваивания =, далее — открывающую *квадратную* скобку, а за ней ввести заданные значения элементов вектора, *отделяя их пробелами или запятыми.* Завершается строка закрывающей квадратной скобкой. Например, ввод строки V = [1.2 -0.3 1.2e-5] задает вектор V, содержащий три элемента со значениями 1.2, -0.3 и 1.2e-5 .(рис.1)

```
» V=[1.2 -0.3 1.2e-5]                                    »
V =

    1.2000   -0.3000    0.0000
```

Рис. I

После ввода вектора система выводит его на экран. То, что в приведенном примере последний элемент выведен как 0, обусловлено установленным форматом Short, в соответствии с которым данные выводятся на экран. Длинный вектор можно вводить частями, которые затем объединять с помощью операции объединения векторов в строку: v = [vl v2] (рис. 2.).

```
» V1= [1 2 3]; V2=[4 5 6];

» V=[V1 V2]
V=
     1    2    3    4    5    6
```

Рис. 2

Язык MatLAB дает пользователю возможность сокращенного ввода вектора, значения элементов которого являются арифметической прогрессией. Если обозначить: nz - начальное значение этой прогрессии (значение первого элемента вектора); kz - конечное значение прогрессии (значение последнего элемента вектора); h - разность прогрессии (шаг), то вектор можно ввести с помощью короткой записи: V = nz : h : kz. Например, ввод строки V = - 0.1 : 0.3 : 1.4 приведет к следующему результату:

```
» V=—0.1:0.3:1.4

v=

   -0.1000    0.2000    0.5000    0.8000    1.1000    1.4000
```

Рис. 3

Если средняя величина (разность прогрессии) не указана, то она по умолчанию принимается равной единице. Например, команда

```
» -2.1 : 5
```

приводит к формированию такого вектора

```
ans=

    Columns I through 7
    -2.1000  -1.1000 -0.1000 0.9000 1.9000 2.9000 3.9000
    Columns 8
    4.9000
```

Таким образом вводятся векторы-строки. Вектор-столбец вводится аналогично, но значения элементов в перечне отделяются знаком ";".

Ввод значений элементов матрицы осуществляется в MatLAB в квадратных скобках по строкам. При этом элементы строки матрицы отделяются друг от друга пробелом или запятой, а строки отделяются друг от друга знаком ";" (рис. 4).

```
» A=[2 4 6 8; 5.5 6.3 6.8 8 8.6]
A=
   2. 000   4.0000   6.0000   8.0000    10.0000
   5.5000   6.3000   6.8000    8.0000     8.6000
```

Рис. 4

2. Формирование векторов и матриц

MatLAB имеет несколько функций, которые позволяют формировать векторы и

матрицы некоторого определенного вида. Перечислим и проиллюстрируем эти функции:
zeros(M,N) - создает матрицу размером (M*N) с нулевыми элементами:

```
» zeros (3,5)
ans=
     0   0   0   0   0
     0   0   0   0   0
     0   0   0   0   0
```

ones(M,N) - создает матрицу размером (M*N) с единичными элементами:

```
» ones (3, 5)
ans =
   1  1  1  1  1
   1  1  1  1  1
   1  1  1  1  1
```

eye(M,N) - создает матрицу размером (M*M) с единицами по главной диагонали и остальными нулевыми элементами:

```
» eye (3,5)
 ans =
    1   0   0   0   0
    0   1   0   0   0
    0   0   1   0   0
```

rand(M,N) - создает матрицу размером (M*N) из случайных чисел, равномерно распределенных в диапазоне от 0 до I:

```
» rand (3,5)
ans =
    0.9501    0.4860    0.4565    0.4447    0.9218
    0.2311    0.8913    0.0185    0.6154    0.7382
    0.6068    0.7621    0.8214    0.7919    0.1763
```

randn(M,N) - создает матрицу размером (M*N) из случайных чисел, распределенных по нормальному закону с нулевым математическим ожиданием и стандартным (среднеквадратическим) отклонением, равным единице:

```
» randn (3, 5)
 ans =
   -0.4326    0.2877    1.1892    0.1746   -0.5883
   -1.6656   -1.1465   -0.0376   -0.1867    2.1832
    0.1253    1.1909    0.3273    0.7258   -0.1364
```

В языке MatLAB предусмотрено несколько функций, которые позволяют формировать матрицу на основе другой (заданной) или используя некоторый заданный вектор. К таким функциям относятся:
fliplr(A) - формирует матрицу, переставляя столбцы известной матрицы относительно вертикальной оси, например:

```
A =
    1    2    3    4    5    6
    7    8    9   10   11   12
   13   14   15   16   17   18
» fliplr (A)
ans =
    6    5    4    3    2    1
   12   11   10    9    8    7
   18   17   16   15   14   13
```

flipud(A) - формирует матрицу, переставляя строки заданной А относительно горизонтальной оси:

```
» flipud (A)
```

ans =

13 14 15 16 17 18
7 8 9 10 11 12
1 2 3 4 5 6

rot90 (A) - формирует матрицу путем "поворота" заданной матрицы A на 90 градусов против часовой стрелки:

» **rot90 (A)**

ans =

6 12 18
5 11 17
4 10 16
3 9 15
2 8 14
1 7 13

reshape (A, m, n) - образует матрицу размером (M*N) путем выборки элементов заданной матрицы A по столбцам и последующего распределения этих элементов по n столбцам, каждый из которых держит m элементов; при этом число элементов матрицы A должно быть равно M*N:

» reshape (A, 2,9)

ans =

1 13 8 3 15 10 5 17 12
7 2 14 9 4 16 11 6 18

tril(A) - образует нижнюю треугольную матрицу на основе матрицы A путем обнуления ее элементов выше главной диагонали:

»**tril (A)**

ans =

1 0 0 0 0 0
7 8 0 0 0 0
13 14 15 0 0 0

triu(A) - образует верхнюю треугольную матрицу на основе матрицы A путем обнуления ее элементов ниже главной диагонали:

»**triu (A)**

ans =

1 2 3 4 5 6
0 8 9 10 11 12
0 0 15 16 17 18

hankel(V) - образует квадратную матрицу Ганкеля, первый столбец которой совпадает с заданным вектором V, например:

» V = [-5 6 7 4]

V =

- 5 6 7 4

» banket (V)

ans =

```
- 5  6  7  4
   6  7  4  0
   7  4  0  0
   4  0  0  0
```

Процедура **diag(x)** - формирует или извлекает диагональ матрицы. Если x - вектор, то функция diag(x) создает квадратную матрицу с вектором x на главной диагонали:

» **diag (V,-1)**

ans =

```
- 5 0 0 0
  0 6 0 0
```

```
   0  0  7  0
   0  0  0  4
```

Чтобы установить заданный вектор на другую диагональ, при обращении к функции необходимо указать еще один параметр (целое число) - номер диагонали (при этом диагонали отсчитываются от главной вверх), например:

»**diag(V,-l)**

```
ans=
      0  0  0  0  0
   - 5  0  0  0  0
      0  6  0  0  0
      0  0  7  0  0
      0  0  0  4  0
```

Если x - матрица, то функция **diag** создает вектор-столбец, состоящий из элементов главной диагонали заданной матрицы x, например, для матрицы A, указанной перед примером применения процедуры **fliplr:**

» diag (A)

```
ans =
     1
     8
    15
```

Если при этом дополнительно указать номер диагонали, то можно получить вектор-столбец из элементов любой диагонали матрицы x, например:

» **diag (A, 3)**

```
ans =
     1
     8
    15
```

Функция **zeros(l, N)** формирует (создает) вектор-строку из N нулевых элементов. Аналогично **zeros(N, 1)** создает вектор-столбец из N нулей.

Векторы, значения элементов которых являются случайными, равномерно распределенными, формируются следующим образом:

rand(l, N) - для вектора-строки и **rand(M, 1)** - для вектора-столбца.

3. Извлечение и вставка частей матриц

Прежде всего отметим, что обращение к любому элементу определенной матрицы в MatLAB осуществляется путем указания после имени матрицы (в скобках, через запятую) двух целых положительных чисел, которые определяют соответственно номера строки и столбца матрицы, на пересечении которых расположен этот элемент.

Пусть имеем некоторую матрицу A:

»**A= [12 3 4; 5 6 7 8; 9 10 11 12]**

```
A =
   1  2  3     4
   5   6   7   8
    9  10  11  12
```

Получить значение элемента этой матрицы, расположенного на пересечении второй строки с третьим столбцом, можно таким образом:

» A (2, 3)

```
ans =
     7
```

Если нужно, наоборот, вставить на это место некоторое число, например pi, то это можно сделать так:

» A (2, 3) = pi; A

```
A =
```

```
   1.0000   2.0000   3.0000   4.0000
   5.0000   6.0000   3.1416   8.0000
   9.000010.0000 11.0000 12.0000
```

Иногда требуется создать меньшую матрицу из большей, формируя ее путем извлечения из последней элементов ее нескольких строк и столбцов, или, наоборот, вставить меньшую матрицу таким образом, чтобы она стала определенной частью матрицы большего размера. Это делается в MatLAB с помощью знака двоеточия (:).

Рассмотрим эти операции на примерах.

Пусть требуется создать вектор VI, который состоит из элементов третьего столбца предыдущей матрицы А. Для этого поступим так:

```
» VI = A (:, 3)
Vl =
    3.0000
    3.1416
   11.0000
```

Создать вектор V2, состоящий из элементов второй строки матрицы А, можно следующим образом:

```
» V2 = A(2,:)
V 2 =
    5.0000    6.0000    3.1416    8.0000
```

Допустим, что необходимо из матрицы А создать матрицу В размером (2*2), состоящую из элементов левого нижнего угла матрицы А. Тогда делают следующее:

```
»B = A(2:3,1:2)
B =
    5    6
    9   10
```

Аналогичным образом можно "вставить матрицу В в верхнюю середину матрицы А:

```
»A(1:2,2:3)=B

A=
    1  2    3    4
    5  9   10   8
    9  10  11  12
```

Для этого вместо указания номеров элементов матрицы можно указывать диапазон изменения этих номеров путем определения нижней и верхней границ, разделяя их двоеточием.

Примечание. Если верхней границей изменения номеров элементов матрицы является ее размер в этом измерении, вместо него можно использовать служебное слово **end**.

Например:

```
» A(2:end, 2: end)
ans =
    9    10    8
    10    11    12
```

Эти операции очень удобны для формирования матриц, большинство элементов которых одинаковы, в частности так называемых разреженных матриц, которые состоят в основном из нулей. Для примера рассмотрим формирование разреженной матрицы размером (5*7) с единичными элементами в центре:

```
» A = zeros(5, 7);
» B = ones(3, 3);
»A(2:4,3:5)=B
 A =
   0  0  0  0  0  0  0
   0  0  1  1  1  0  0
   0  0  1  1  1  0  0
   0  0  1  1  1  0  0
   0  0  0  0  0  0  0
```

"Растянуть" матрицу (A) в единый вектор (V) можно с помощью простой записи V = A(:). При этом создается вектор-столбец с количеством элементов (M*N), в котором столбцы исходной матрицы размещены сверху вниз в порядке самих столбцов:

```
» A = [ 1 2 3 ; 4 5 6 ]
A =
    1    2    3
    4    5    6
» v = A(:)
 v =
    1
    4
    2
    5
    3 6
```

Наконец, "расширять" матрицу, составляя ее из отдельных заданных матриц ("блоков") можно тоже довольно просто. Если заданы несколько матриц-блоков Al, A2, ..., AN с одинаковым числом строк, то из них можно "слепить" единую матрицу A, объединяя блоки в одну "строку" таким образом:

A = [A1, A2, ..., AN]

Эту операцию называют горизонтальной *конкатенацией* (сцеплением) матриц. Аналогично, вертикальную конкатенацию матриц можно реализовать при условии, что все составляющие блоки-матрицы имеют одинаковое число столбцов, применяя для отделения блоков вместо запятой — точку с запятой:

A = [A1; A2;...; AN]

Пример горизонтальной конкатенации:

```
» A1 = [1 2 3; 4 5 6; 7 8 9];
» A2 = [10; 11; 12);
» A3 - [14 15; 16 17; 18 19];
»A = [A1,A2,A3]
A =
    1    2    3   10   14   15
    4    5    6   11   16   17
    7    8    9   12   18   19
```

Пример вертикальной конкатенации:

```
» B1 = [1 2 3 4 5];
» B2 = [6 7 8 9 10; 1112 13 14 151;
» B3 - (17 18 19 20 21);
» B = [Bl; B2; B3]
B =
    1    2    3    4    5
    6    7    8    9   10
   11   12   13   14   15
   17   18   19   20   21
```

4. Действия под векторами

Выделим две существенно различающиеся группы действий над векторами: векторные действия — такие, которые предусмотрены векторным исчислением в математике; действия по преобразованию элементов — это действия, которые преобразуют элементы вектора, но не являются операциями, разрешенными математикой с векторами.

Векторные действия над векторами

Сложение векторов. Как известно, складываться (суммироваться) могут только векторы одинакового типа; т.е. оба вектора должны быть либо векторами-строками, либо векторами-столбцами и иметь одинаковую длину (одинаковое число элементов). Если X и Y именно такие векторы, то их сумму Z можно получить, введя команду Z = X + Y. например:

```
»x = [ 1 2 3 ];    y = [4 5 6];
» v=x + y
v =
     5    7    9
```

Аналогично с помощью арифметического знака "-" осуществляется вычитание векторов, имеющих одинаковую структуру:

```
» v = x - y

v =
   -3   -3   -3
```

Транспонирование вектора осуществляется применением знака апострофа, который

записывается сразу за записью имени исходного транспонируемого вектора:

»x'

ans =

1
2
3

Умножение вектора на число осуществляется в MatLAB с помощью знака арифметического умножения (*) таким образом: Z = X*г или Z = г*X, где г — некоторое действительное число.

» v= 2 * x

V =

2 4 6

Умножение двух векторов определено в математике только для векторов одинакового размера (длины) и лишь тогда, когда один из векторов-сомножителей является строкой, а второй — столбцом. Иначе говоря, если векторы X и Y являются строками, то математический смысл имеют только две формы умножения этих векторов: U = X' * Y и V = X * Y'. Причем в первом случае результатом будет квадратная матрица (диада), а во втором — число. В MatLAB умножение векторов осуществляется с применением обычного знака умножения (*), который записывается между сомножителями-векторами.

Пример:

»x = [1 2 3]; y=[4 5 6];

» v=x' * y

v =

4 5 6
8 10 12
12 15 18

» v =x * y'

v =

32

Для трехкомпонентных векторов в MatLAB существует функция **cross,** которая позволяет найти векторное произведение двух векторов. Если заданы два трехкомпонентных вектора vl и v2, то для этого достаточно ввести оператор:

» cross (vl, v2)

Пример:

»vl =[123|; v2 = [456];

» cross (vl, v2)

ans =

-3 6 -3

На этом перечень допустимых математических операций с векторами исчерпывается.

Действия по поэлементному преобразованию векторов

В языке MatLAB есть ряд операций, которые преобразуют заданный вектор в другой того же размера и типа, но в то же время не являются математическими операциями с вектором как математическим объектом. Все эти операции преобразуют элементы вектора как элементы обычного одномерного массива чисел. К таким операциям принадлежат, например, все элементарные математические функции, приведенные в лаб. 2., которые зависят только от одного аргумента. В языке MatLAB запись вида Y = sin(X), где X — некоторый известный вектор, приводит к формированию нового вектора Y, имеющего тот же тип и размер, но элементы которого равны синусу соответствующих элементов вектора-аргумента X. Приведем примеры:

»x = [-2,-1,0,1,2];

» y = sin (x)

y=

-0.9093 -0.8415 0 0.8415 0.9093

» z = tan (x)

z=

2.1850 -1.5574 0 1.5574 -2.1850

» v = exp (x)

v =

0.1353 0.3679 1.0000 2.7183 7.3891

Кроме этих операций в MatLAB предусмотрено несколько операций поэлементного преобразования, осуществляющихся с помощью знаков обычных арифметических действий. Все эти операции применяются к векторам одинакового типа и размера. Результатом их является вектор того же типа и размера.

Добавление (вычитание) числа к (из) каждому(го) элементу (а) вектора. Осуществляется с помощью знака "+" ("-").

Поэлементное умножение векторов. Производится при помощи совокупности знаков ".*", которая записывается между именами перемножаемых векторов. В результате получается вектор, каждый элемент которого является произведением соответствующих элементов векторов- "сомножителей".

Поэлементное деление векторов.

Осуществляется при помощи совокупности знаков "./". Результат — вектор, каждый элемент которого является частным от деления соответствующего элемента первого вектора на соответствующий элемент второго вектора.

Поэлементное деление векторов в обратном направлении. Осуществляется с помощью совокупности знаков ".\". В результате получают вектор, каждый элемент которого является частным от деления соответствующего элемента второго вектора на соответствующий элемент первого вектора.

Поэлементное возведение в степень. Осуществляется при помощи совокупности знаков ".^". Результат — вектор, каждый элемент которого является соответствующим элементом первого вектора возведенным в степень, величина которой равна значению соответствующего элемента второго вектора. Пример:

» x = [1,2,3,4,5]; y = [-2,1,4,0,5];

» **disp (x+2)**

3 4 5 6 7

» **disp (y-3)**

-5 - 2 1 - 3 2

» **disp (x.*y)**

-2 2 12 0 25

» disp (x./y)

Warning: Divide by zero.

-0.5000 2.0000 0.7500 Inf 1.0000

» **disp (x.\y)**

-2.0000 0.5000 1.3333 0 1.0000

» **disp (x.^y)**

1 2 81 1 3125

Вышеуказанные операции позволяют очень просто вычислять значения (а затем -строить графики) сложных математических функций, не используя при этом операторы цикла, т.е. производить построение графиков в режиме калькулятора. Для этого достаточно задать значения аргумента как арифметическую прогрессию так. как это было показано в начале работы, а затем записать нужную функцию, используя знаки поэлементного преобразования векторов.

Например, пусть требуется вычислить значения функции:

$$y = a \cdot -e^{-hx} \cdot \sin x$$

при значениях аргументах от 0 до 10 с шагом 1. Вычисление массива значений этой функции в указанных условиях можно осуществить при помощи только двух простых операторов:

» a = 3; h - 0.5;

»x = 0: 1 : 10;

» y = a * exp(-h * x) .* sin(x)

y =

0 1.5311 1.0035 0.0945 -0.3073 -0.2361

-0.0417 0.0595 0.0544 0.0137 -0.0110

5. Поэлементное преобразование матриц

Для поэлементного преобразования матрицы пригодны все указанные ранее во второй работе алгебраические функции. Каждая такая функция формирует матрицу того же размера.

что и заданная, каждый элемент которой вычисляется как указанная функция от соответствующего элемента заданной матрицы. Кроме того, в MatLAB определены операции поэлементного умножения матриц одинакового размера (с помощью символов ".*", помещаемых между именами "перемножаемых" матриц), поэлементного деления (символов "./" и ".\"), поэлементного возведения в степень (символов ".^"), когда каждый элемент первой матрицы возводится в степень, равную значению соответствующего элемента второй матрицы.

Приведем несколько примеров:

```
» A = [1, 2,3, 4, 5; -2, 3,1, 4, 0]
A =
    1   2   3   4   5
   -  2   3   1   4   0
» B = [-1, 3, 5, -2,1; 1, 8, -3, -1, 2]
B =
   -1  3   5  -2   1
    1  8  -3 -1    2
»sin(A)
ans =
   0.8415 0.9093 0.1411 -0.7568 -0.9589
  -0.9093 0.1411 0.8415 -0.7568      0
»A .* B
ans=
-1   6  15  -8   5
-2  24 –3 -4   0
```

```
» A./ B
ans =
   -1.0000    0.6667    0.6000   -2.0000    5.0000
   -2.0000    0.3750   -0.3333   -4.0000         0
» A.\B
Warning: Divide by zero.
ans =
 -1.0000 1.5000 1.6667 -0.5000 0.2000
   -0.5000 2.6667 -3.0000 -0.2500  Inf
» A.^ B
ans =
  l.0e+003*
   0.0010 0.0080 0.2430 0.0001 0.0050
   -0.0020 6.5610 0.0010 0.0003   0
```

Оригинальной в языке MatLAB является операция прибавления к матрице числа. Она записывается таким образом: A +x или x + A (где A — матрица, а x — число). Такой операции нет в математике. В MatLAB она эквивалентна совокупности операций:

$$A+x*E$$

где E — обозначение матрицы тех же размеров, что и матрица A, состоящей только из единиц.

Например:

```
»A=[12 3 4 5; 67 8 9 11]
A=
   1   2    3   4   5
   6   7    8   9   1
»A+2
ans =
   3   4   5   6   7
   8   9   10   11   13
»2+A
ans =
    3   4   5   6   7
   8   9   10   11   13
```

6. Матричные действия с матрицами

К матричным действиям с матрицами относятся такие операции, которые используются в матричном исчислении в математике.

Базовые действия с матрицами — сложение, вычитание, транспонирование, умножение матрицы на число, умножение матрицы на матрицу, возведение матрицы в целую степень — осуществляются в языке MatLAB с помощью обычных знаков арифметических операций. При использовании этих операций важно помнить условия, при которых эти операции возможны:

- при сложении или вычитании матриц они должны иметь одинаковые размеры;
- при умножении матриц число столбцов первого множителя должно совпадать с числом строк второго множителя.

Невыполнение этих условий приведет к появлению в командном окне сообщения об

ошибке.

Приведем несколько примеров.

Пример сложения и вычитания

```
» A  =  [12345;678911]
A =
    1   2   3   4   5
    6   7   8   9   11
» B = ( -1 -2 -3 -4; 5 6 7 8 9]
B =
    0   -1   -2   - 3 - 4
    5   6   7   8   9
» A + B
ans =
    1   1   1   1   1
   11    13   15   17   20
» A - B
ans =
    1   3   5   7   9
    1   1   1   1   2
```

Пример умножения на число

```
» 5 * A
ans=
    5  10  15  20  25
   30  35 40  45  55
» A * 5
ans =
    5   10   15   20   25
   30   35   40   45   55
```

Пример транспонирования матрицы

```
» A'
ans =
    1  6
    2  7
    3  8
    4  9
    5  11
```

Пример умножения матрицы на матрицу

```
» A' * B
ans =
    30   35  40   45   50
   35    40  45   50   55
   40    45  50   55   60
   45    50  55   60   65
   55    61  67   73   79
» C = A*B'
C =
   -40   115
   -94   299
```

Функция обращения матрицы — inv(A) — вычисляет матрицу, обратную заданной матрице A. Исходная матрица A должна быть квадратной, и ее определитель не должен быть равен нулю. Приведем пример:

» inv (C)

```
ans =
    -0.2600    0.1000
    -0.0817    0.0348
```

Проверим правильность выполнения операции обращения, применяя ее еще раз к полученному результату:

» inv (ans)

```
ans =
  -40.0000
  115.0000
  -94.0000
  299.0000
```

Как видим, мы получили исходную матрицу C, что подтверждает правильность выполнения обращения матрицы.

Возведение матрицы в степень осуществляется в MatLAB при помощи знака "^" (например, A^ n). При этом n должно быть целым (положительным или отрицательным) числом. Это матричное действие эквивалентно умножению матрицы A на себя n раз (если n — положительно) либо умножению обратной матрицы на себя (при n отрицательном).

Весьма оригинальными в языке MatLAB являются две новые, неизвестные в математике функции деления матриц. При этом вводятся понятия деления матриц слева направо и деления матриц справа налево. Первая операция записывается при помощи знака "/", а вторая — при помощи знака "\", которые помещаются между именами матриц, которые делятся друг на Друга.

Операция B/A эквивалентна такой последовательности действий: B * inv(A). Здесь функция **inv** осуществляет обращение матрицы. Ее удобно использовать для решения матричного уравнения:

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

Аналогично операция A\B равносильна совокупности операций **inv(A)*B**, которая является решением матричного уравнения:

$$\boldsymbol{A}\mathbf{*X = B}$$

Для примера рассмотрим задачу нахождения корней системы линейных алгебраических уравнений:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 14 \\ 2 & \\ 2x_1 - x_2 - 5x_3 &= -15 \\ x_1 - x_2 - x_3 &= -4 \end{aligned}$$

В системе MatLAB это можно сделать таким образом:

»A=[1 2 3; 2-1-5; 1-1-1]

```
A =
     1    2    3
     2   -1   -5
     1   -1   -1
```

»B = [14;-15;-4]

```
 B =
    14
   -15
    -4
```

» x = A \ B

```
x =
```

1

2

3

7. Матричные функции

Вычисление матричной экспоненты *(e^4)* производится с помощью функций **expm, expml, expm2 и expm3.** Эта функции следует отличать от ранее упомянутой функции **exp(A),** формирующей матрицу, каждый элемент которой равен *e* в степени, равной соответствующему элементу матрицы А.

Функция **expm(A)** является встроенной функцией MatLAB. Функция **expml(A)** является М-файлом. вычисляющим матричную экспоненту путем использования разложения Паде матрицы А. Функция **expm2(A)** вычисляет матричную экспоненту, используя разложение Тейлора матрицы А. Функция **expm3(A)** вычисляет матричную экспоненту, используя спектральное разложение А.

Приведем примеры использования этих функций:

```
»A = [1,2,3;0,-1,5;7,-4, 1]
A =
    1     2     3
    0   - 1     5
    7   - 4     1
```

```
» expm (A)
ans =
  131.3648   -9.5601   80.6685
   97.8030   -7.1768   59.9309
  123.0245   -8.8236   75.4773
» expml (A)
ans =
  131.3648   -9.5601   80.6685
   97.8030   -7.1768   59.9309
  123.0245   -8.8236   75.4773
```

```
» expm2 (A)
ans =
  131.3648   -9.5601   80.6685
   97.8030   -7.1768   59.9309
  123.0245   -8.8236   75.4773
» expm3 (A)
ans =
  l.0e+002*
   1.3136 + 0000i -0.0956 – 0.0000i  0.8067 – 0.0000i
   0.9780 + 0.0000i -0.0718 + 0.0000i  0.5993 – 0.0000i
   1.2302 +0.0000i -0.0882 – 0.0000i  0.7548 – 0.0000i
```

Функция **logm(a)** производит обратную операцию — логарифмирование матрицы по натуральному основанию, например:

```
»A= [1 2 3; 0 1 5; 7 4 1]
A =
    1     2     3
    0     1     5
    7     4     1
» B = expm3 (A)
B =
  l.0e+003*
   0.9378   0.7987   0.9547
   1.0643   0.9074   1.0844
   1.5182   1.2932   1.5459
»logm (B)
ans =
   l.0000    2.0000    3.0000
   0.0000    1.0000    5.0000
   7.0000    4.0000    1.0000
```

Функция **sqrtm(A)** вычисляет такую матрицу Y, что Y*Y =

A:

» Y = sqrtm (A)

```
Y =
   0.7884 + 0.8806i 0.6717 - 0.1795i 0.8029 - 0.4180i
   0.8953 + 0.65081 0.7628 + 0.86201 0.9118-
   1.00661 1.2765 - 1.40921 1.0875-0.54491
   1.3000+1.25251
```

»Y*Y'

```
ans =
   1.0000 - 0.0000i  2.0000 + 0.0000i  3.0000 - 0.0000i
   0.0000 + 0.0000i  1.0000 + 0.00001  5.0000-0.0000i
   7.0000 +0.0000i   4.0000 + 0.0000i  1.0000 - 0.0000i
```

Лабораторная работа № 3

1.1.1. Арифметика в MATLAB'е

В MATLAB'е есть основные арифметические операции: + (сложение), - (вычитание), * (умножение) и / (деление). Степень обозначается через ^, так что набрав

```
» 5*5+12^2
```

и нажав < Enter >, получим

```
ans =

     169
```

Не забывайте нажимать <Enter> после набора строки, чтобы послать ее для выполнения. Законы старшинства операций встроены, но в сомнительных случаях пользуйтесь круглыми скобками. Например, для строки

```
» 8*(1/(5-3)-1/(5+3))
ans =
     3
```

Элементарные функции, известные вам по работе с ручными калькуляторами, здесь также реализованы. Выполните строки

```
» sqrt(5^2+12^2)
```

и

```
» exp(log(1.7))
```

А что, думаете, даст sin (pi/2)? Попробуйте.

На самом деле MATLAB имеет для числа *pi* встроенное значение pi = 3.1415926... Просто наберите pi, когда оно вам потребуется. Попробуйте следующее:

```
» pi
» format long
» pi
» format short
```

MATLAB выведет значительно больше значащих цифр, чем выдается по умолчанию в режиме *format short.*

1.1.2. Использование переменных

Вы можете приписать числовое значение « *переменной*» для использования в последующих вычислениях. Выполните

```
» x=3
```

и получите

```
x =
     3
```

Но можно получить и что-нибудь более полезное, например,

```
» rad=2; ht=3;
» vol=pi*ht*rad^2

vol =
```

37.6991

Обратите внимание, что первая строка содержит две «команды» и ни одна из них не выдает результата! Когда MATLAB встречает инструкцию с символом ; (точка с запятой) в конце, он запрещает вывод результата. Инструкция в действительности выполняется, но ее результат умалчивается, что вы можете проверить, выполнив

```
» rad=4;
» rad

rad =

     4
```

Использование символа ; позволяет избежать хаотического заполнения экрана промежуточными результатами. Помните об этом, разбирая последующие примеры.

Не забывайте, что каждая переменная должна как-то получить значение прежде, чем вы сможете использовать ее в дальнейших вычислениях. Так, после выполнения предыдущих примеров и строки

» f = x^2 + 2*x*y + y^2 будет выдано примерно такое сообщение

??? Undefined function or variable y

Это не требует пояснений. После засылки y=4; повторное вычисление f пройдет успешно.

Кстати отметим, что быстрый способ повторить предыдущую строку MATLAB'a —это нажимать клавишу «стрелка-вверх» до тех пор, пока не выберется желаемая команда. Попробуйте это сейчас. Если исходная строка была не совсем правильной или же вы хотите получить новую строку из сложной, но схожей с ней и выполненной ранее, вы можете воспользоваться этим же приемом. Выбрав требуемую строку, используйте «стрелки-в-сторону» совместно с клавишей Delete, чтобы отредактировать ее нужным образом. В качестве упражнения попробуйте, используя предыдущие строки, вычислить объем правильного круглого цилиндра с радиусом 2 и высотой 1/4. Вы получили pi?

При затруднениях с запоминанием уже заданных вами имен переменных попробуйте выполнить who или whos. Попробуйте обе команды. Вы узнаёте перечисленные переменные?

1.2. Векторы и графики

Одно из удовольствий, которое вы будете испытывать при изучении MATLAB'a, состоит в простоте построения графиков. Основные принципы таковы:

(i) выберите последовательность x-значений, т.е. вектор значений аргумента;
(ii) вычислите *y* = /(x), т.е. получите соответствующий вектор *y*-значений;
(ш) нарисуйте график *y* от *x*.

Прежде чем проделать это, стоит потратить немного времени, чтобы узнать кое-что о том, как MATLAB работает с векторами.

1.2.1. Векторы

Выполните следующие примеры, в которых все результаты будут векторами. Не спешите, обдумывая каждый результат.

```
» и=[2,2,3]
» и=[2 2 3]
» v=[1,0,-1]
» w=u-2*v
» range=1:13
```

```
» odd=l:2:13
» down=20:-0.5:0
» even=odd+l
» xgrid=0:.05:1; x=xgrid*pi
» y=sin(x)
```

Первые две строки показывают, что элементы вектора могут разделяться пробелами или запятыми. Если вы боитесь вставить пробел случайно, то можете придерживаться записи с запятыми. Таким образом, [1+1 2 3] означает то же, что и [2,2,3], а [1 +1 2 3] — то же, что и [1,1,2,3]!

Заметьте, что векторы могут быть любой длины. Они могут быть строками, как выше, или векторами, подобными

```
» w'
ans =
     0
     2
     5
```

где апостроф обозначает транспонирование (T). В MATLAB'е векторы трактуются просто как специальный случай матриц, о которых вы узнаете значительно больше в следующей главе.

Обратите внимание, что произошло, когда представляемый вектор оказался слишком длинным и не уместился в одной строке. Тогда система сначала отображает столько элементов, сколько их умещается в строке, а остальные переносит на следующие строки. Элементы вектора-строки трактуются как «столбцы».

Элементарная функция вектора x, такая, как sin(x), также является вектором того же типа. Мы можем использовать этот факт при создании графиков функций, как показано в следующем разделе.

MATLAB знает как перемножать матрицы соответствующих размеров. Это будет обсуждаться подробнее в следующей главе. А сейчас попробуйте выполнить строки

```
» w*w'
» w'*w
» u*w'
» u*u
```

Вы понимаете смысл полученных результатов? Почему последняя строка не работает?

Пусть теперь вы хотите получить множество значений z, данное выражением $z=y^2$, где вектору y уже были присвоены некоторые значения. Из предыдущего опыта вы понимаете, что

```
» z=y*y
```

не приемлемо для MATLAB'а. Присвоение

```
» z=y*y'
```

выполняется системой, но вычисляется как скалярное произведение y • y! Чтобы заставить MATLAB перемножить векторы *поэлементно*, выполните

```
» z=y.*y
```

где точка перед символом * есть ключевой признак поэлементной операции. Подобным же образом u./v и y.^2 понимаются как поэлементные операции над векторами одинаковых размеров.

1.2.2. Кое-что о графиках

Теперь выполните whos, чтобы удостовериться, что x и y определены, как выше. Они оба должны быть 1 x 21-матрицами (т.е. векторами-строками).

Построить график легко. Просто выполните

```
» plot(x,y)
```

и чуть подождите. Как по волшебству, появится замечательная незамысловатая кривая $y = sinx$ от аргумента x. Оси выбираются автоматически в соответствии с областями изменения

переменных. Это простейший возможный случай. Потом вы захотите делать более сложные вещи. А сейчас попробуйте следующее:

```
» title(‘Graph of y=sin(x)')
» xlabel('x')
» ylabel('y')
» yl=2*x;
» hold on
» plot(x,yl,'r')
```

Вероятно, вы сможете разгадать значение каждой из этих команд. Например, yl=2*x определяет значения новой функции *y = 2x,* hold on дает MATLAB'y указание сохранить выведенный график, а plot(x,yl, 'r') рисует новую кривую поверх прежней. Заметьте, что оси были скорректированы и вторая кривая нарисована красным цветом.

В этих примерах соседние точки соединялись прямолинейными отрезками. Если захотите, вы можете задать вид точек, которым рисуется кривая, выбрав символы, как показано далее. Выполним строки

```
» hold off
» plot(x,y,'+')
» plot(x,y,'g*')
» plot(x,y,'w.')
```

MATLAB сделал то, что вы ожидали? Вы воспользовались клавишей !, чтобы повторно выполнить предыдущие команды? Можно получить подсказку об использовании любой команды MATLAB'a.

Выполните, например,

```
» help plot
» help hold
» help sin
```

и т.д.

1.3. Создание и редактирование скрипт-файлов

Начав работать, вы вскоре найдете утомительным снова и снова вводить те же самые или подобные им последовательности команд. К счастью, есть простой путь обойти это: нужно просто сохранить любую часто повторяемую последовательность команд в виде файла, называемого *«скриптом»* или *«М- файлом»*. После этого можно вызывать этот список команд так часто, как надо.

Например, в каком-то сеансе работы вы захотели найти расстояние между точками *A* и *B,* заданными соответственно векторами a = (1, 0, —2) и b = (2, 3, 1). Зная, что вектор смещения между ними равен

$$d = b - a$$

и что

$$!d!^2 = d*d$$

вы можете воспользоваться следующей последовательностью команд MATLAB 'a:

```
» a=[1,0,-2];
» b=[2,3,1];
» d=b-a;
» dd=d*d';
» dist=sqrt(dd)
```

чтобы решить эту частную задачу. Это неплохо, но, предположим, у вас теперь пять точек, и

нужно выбрать из них две наиболее близкие друг к другу. Очевидно, что тогда вы захотите сохранить в «скрипте» (файле) как можно больше таких шагов, которые допускали бы затем их полное повторение.

1.3.1. Редактирование и сохранение текстовых файлов

Сначала нам необходимо рассмотреть управление файлами и их редактирование.

Пользователи, не использующие Windows

Если вы *не* используете Microsoft Windows, то здесь вам нужно будет сделать некоторые изменения в процедурах. Однако как бы ни был MATLAB инсталлирован на вашем компьютере, без сомнения там будет какой-нибудь *текстовый редактор*. Предполагая, что он носит имя edit, вероятно самый легкий способ вызвать его из MATLAB'a — это выполнить

```
» !edit fname
```

где fname —имя текстового файла, который либо уже существует, либо будет существовать ко времени окончания вами работы. Если это не срабатывает, проконсультируйтесь с кем-то, знающим настройку вашей системы, или с более опытным пользователем.

Пользователи Windows

Windows устанавливается со своим собственным базовым редактором текстовых файлов под именем Блокнот *(Notepad),* пиктограмму которого можно обычно найти в разделе Стандартные программы (Accessories Group). Типичная установка MATLAB'a в Windows прямо использует эту программу, почему и мы ограничимся ею. Чтобы открыть и отредактировать новый файл с именем myfile.m, *непосредственно из MATLAB'a,* сделайте следующее:

(i) В меню MATLAB Command Window щелкните мышью на File.

(ii) Щелкните на New, а затем на M-f ile.

(iii) В Блокноте, который вы только что открыли, можно набирать любые строки, например,

```
% myf ile.m
% Это просто название (идентификатор) самого файла..
'% Эти три строки суть строки комментария, на которые
% MATLAB не обращает внимания.
disp (‘I am an M-f ile')
```

(iv) Щелкните на File а затем на Save As.

(v) В окошечке File Name, открывшемся в ожидании, наберите myfile.m.

(vi) Щелкните на OK.

Вы только что создали файл, который MATLAB может найти и использовать.

Вернувшись в MATLAB Command Window, вы теперь можете спросить MATLAB, найдет ли он этот файл. Выполните

```
» type myfile
```

и увидите строки, которые набрали раньше. Если этого не получилось, вернитесь к шагу (i) и снова вызовите Блокнот щелчком по File в MATLAB Command Window, но дальше выберите Open M-file. Вы увидите окно ввода, соответствующее myfile .m. Если не получилось и это, вернитесь прямо к началу этого раздела.

1.3.2. Скрипт-файлы

Если все прошло нормально, то теперь вы имеете первый прш скрипт-файла. Чтобы использовать его, просто наберите

```
» myfile
```

после чего увидите что-то вроде

```
I am an M-file
```

Теперь о чем-то более полезном. Откроем М-файл, чтобы повторить ранее приведенные команды для определения расстояния между двумя точками. Действуя, как и раньше, откройте

новый М-файл с именем distab.m, содержащий несколько строк с комментариями

```
%  distab.m
%  Вычисляет расстояние между двумя векторами a и b
%  …
```

включая дополнительные строки (те, которые начинаются с %), которые помогут вам вспомнить, как это работает. Далее пойдут рабочие строки

```
d=b-a;
dd=d*d';
dist=sqrt(dd)
```

Не забудьте сохранить файл, щелкнув мышью сначала по File, а затем по Save As, как это делалось при создании mfile.m. Если необходимо, еще раз просмотрите тот пример. Окончив редактирование файла, при желании можно закрыть Блокнот, щелкнув по File и затем по Exit. Можно оставить Блокнот и открытым, но это может привести к путанице, если окажется, что открыто слишком много окон с Блокнотом. При закрытии Блокнот всегда сам напомнит вам о сохранении последних изменений в файле.

Теперь задайте компоненты векторов а и Ь, если не сделали этого раньше:

```
» a=[1,0,-2];
» b=[2,3,1];
```

Затем найдите расстояние, просто выполнив

```
» distab
```

Получилось? Если нет, вернитесь в Блокнот и сделайте еще одну попытку.

Проделайте это для различных точек Л и В, в частности и для таких пар, где правильность определения расстояния просто проверить, например, для A = (1,2,3) и B = (1,1,3).

```
» a=[1,2,3];
» b=[1,1,3];
» distab
```

Чтобы узнать, какие М-файлы вы создали или какие еще есть в MATLAB'e, воспользуйтесь командой what. Если же вы хотите проверить назначение М-файла, можно использовать команду help (точно так же, как для любой команды MATLAB'a):

```
» help myfile
» help sqrt
» help sin
```

Команда help выводит начальные строки с *комментариями* в начале М-файла. Вот почему всегда считается *хорошим правилом* включать строки комментария (начинающиеся с %) в шапку М-файла. Также полезно включать в первую строку комментария имя файла. Заметьте, что myfile —это имя инструкции MATLAB'a (того, что вы набираете для исполнения), тогда как myfile.m *имя файла*, содержащего ее определение.

1.3.3. Файлы функций

Было бы утомительным присваивать значения компонентам двух векторов перед каждым обращением к нашему «скрипту». Можно соединить присвоение входных значений с действующими инструкциями вызова М-файла, используя М-файл типа *функция*. Кроме того, одновременно можно присвоить полученные значения новым переменным, т.е. создать файл-функцию distfn, такую, что, набрав

```
» dab=distfn([l,2,3],[1,1,3]);
```

или

```
» a=[1,2,3];  b=[1,1,3]);
» dab=distfn(a,b);
```

вы присвоите правильное значение расстояния переменной dab без каких-либо дополнительных хлопот.

Вот как нужно сделать изменения в скрипт-файле distab.m, чтобы он стал файлом-функцией distfn.m. Пусть ваш Блокнот уже закрыт, так что начнем редактирование с самого начала:

(i) В меню MATLAB Command Window щелкните мышью на File.
(ii) Щелкните по Open M-file, чтобы открыть Блокнот, и просмотрите список доступных М-файлов,'т.е. оканчивающихся на . т. (если потребуется, смените окончание *. txt в рамке ввода имени файла File Name на * .m).
(iii) Выберите distab.m.
(iv) Теперь можно внести изменения. Сначала измените строки комментария, чтобы отразить новое имя и назначение файла, а затем измените исполняемые инструкции MATLAB'a так. чтобы ваш файл принял вид:

```
% distfn.m
%Вычисляет расстояние между двумя точками, заданным;'
% векторами a и b
% Вызов:
% dist=distfn(a,b)
% ввод: a,Ь (векторы положения точек)
% вывод: distfn - расстояние между точками
function dist=distfn(a,b)
d=b-a;
dd=d*d';
dist=sqrt(dd);
```

(v) Щелкните по Save As.
(vi) В окошечке File Name, открывшемся в ожидании, наберите distfn.m.
(vii) Щелкните по OK.

Вернувшись в MATLAB Command Window, вы теперь можете, набрав help distfn, прочесть, как пользоваться этой функцией. Выполните

```
» dist=distfn([l,l,l],[2>2,2])
```

или

```
» dist=distfn(a,b)
```

чтобы присвоить найденное расстояние переменной dist. Если вы вносите не одно исправление, то можете забыть поставить в некоторых местах символы ; . Для исправления любых ошибок просматривайте файл, используя Блокнот.

М-файлы-функции (или «М-функции»), т.е. М-файлы с первой исполняемой строкой вида function ..., обладают одним очень важным свойством. Кроме самого имени (здесь это distf n), все другие переменные (a, Ъ, dd и т.д.) — чисто внутренние для этой функции. Это помогает уменьшить путаницу с другими вычислениями и переменными, используемыми вами. Проверьте это, выполнив

```
» who
» clear
```

» who

» dist=distfn([l,l,l],[2,2,2])

» who

Команда clear удаляет все ранее определенные переменные. После выполнения функции dist=distfn(..) внутренние переменные не оставляют и следа в вашем MATLAB-сеансе.

1.3.4 Файлы дневника работы и сохранение файлов

Иногда вам может понадобиться запомнить те ваши действия, которые отображались на экране дисплея. Позднее вам может потребоваться распечатать их (см. следующий раздел). Сделать это очень легко. Чтобы понять это, выполните

» diary sectl.txt

» % Начало раздела 1

» % сейчас несколько команд
» myfile
» dist=distfn([l,l,l])[2,2,2])
» diary sect2.txt
» % Я хочу сохранить это где-то еще
» x=0:.1:1;
» y=x.*x
» plot(x.y)
» diary off

Сеанс будет протекать вполне нормально. Команда diary fname дает MATLAB'у указание записать копию выводимого на экран текста (числа и буквы) в файл с именем fname. Команда diary off прекращает запись. В приведенном сейчас примере часть вывода записывается в один файл, а часть — в другой. Для просмотра полученного вы можете воспользоваться Блокнотом и, если необходимо, отредактировать записи. Вызовите Блокнот, как обычно щелкнув по File, и с помощью Open просмотрите список файлов с окончаниями имен на *.txt. В этом списке вы увидите sectl.txt и sect2.txt.

Взглянув на sect2.txt, вы, вероятно, будете разочарованы, не найдя там графика $y = x^2$. Это потому, что графики и другие графические образы нелегко представить в текстовой форме. В дальнейшем будет показано, как обойти эту трудность.

При окончании сеанса с MATLAB'ом все текущие переменные и их значения теряются. Обычно это не создает проблем. Начать сеанс заново можно строкой

» clear

освободившись тем самым от всего предыдущего. Если же вы все-таки хотите сохранить то, что получили, наберите, например,

» save monday

или

» save monday x,y

где вторая команда сохранит только явно указанные переменные в файле monday. Можно вновь загрузить сохраненное на следующий день или когда пожелаете командой

» load monday

Помните, что таким способом сохраняются лишь сами переменные, т.е. их текущие значения. Все формулы, которые вы применяли, будут потеряны, если вы не запомните их в некотором М-файле.

1.4. Получение распечаток

1.4-1- Пользователи Windows

Одно из преимуществ использования среды Windows состоит в том, что печать текстовых файлов, таких, как файлы дневника работы или М-файлы, выполняется одинаково во всех приложениях MATLAB — стандартное приложение и поэтому использует все достоинства такого подхода. Это же верно и для графиков, полученных с помощью MATLAB'а.

Чтобы получить печатную копию М-файла или другого текстового файла, просто откройте его с помощью Блокнота обычным образом (см. предыдущий раздел). Щелкните на File, а затем на Print. И все! Если не получилось, возможно, вам надо проверить Print Setup в том же самом меню, чтобы узнать, куда выводится распечатка. При необходимости вы можете вызвать вашего местного специалиста по обслуживанию компьютеров. Но обычно все уже настроено, так что все распечатки идут на нужный принтер.

Графики выводятся на печать точно так же, но вы должны использовать кнопки File и Print в окне, в котором построено изображение (обычно это Figure No. 1).

1.4-2. Пользователи, не работающие под Windows

Если у вас нет системы Windows или вы хотите обойтись без нее, можно выполнить

```
» !print fname.txt
```

или воспользоваться подходящей командой печати вашей операционной системы (например, 1p или 1pr для Unix).

Для графиков команда MATLAB'a print обычно настроена на вывод текущего изображения на принтер, заданный по умолчанию. Если это не так, попробуйте воспользоваться советами из help print. Потерпев неудачу и здесь, расспросите вашего терпеливого друга.

Таблица 1.1. Сводка вышеприведенных основных команд

plot (x, y)	plot(x,y,'*')	plot(x,y,'+g')
title	zlabel	ylabel
('Заголовок')	('Надпись по оси x')	('Надпись по оси y')
sqrt (x)	sin(x)	exp(x)
hold on	hold off	
x=-1:.2:1	y=x . *x	dotprod=x*y'a)
format long	format short	
help sqrt	help myfile	save fname
diary filel.txt	diary off	load fname

(' = транспонирование)

Упражнение. Если вы не сможете вспомнить некоторые из команд, загляните в табл. 1.1. При необходимости вернитесь назад и перечитайте нужный раздел.

1.1 Найдите сумму первых четырех членов последовательности

1.2 Определите вектор t со значениями компонент, равномерно расположенными с шагом 0.2 между 0 и 6 включительно. Теперь используйте его, чтобы нарисовать кривые

$$f(t) = \sin(\pi t) \qquad g(t) = \exp(-t)\sin(\pi t)$$

на одном графике, изобразив первую зеленым, а вторую — желтым цветом Если вы не уверены, как применить нужные вам функции MATLAB'a, наберите help exp и т.д. Улучшите график, добавив белую линию, соответствующую y = 0.

1.3 Воспользуйтесь редактором, чтобы создать М-файл, в котором определяется длина каждой из сторон треугольника ABC, вершины которого заданы векторами a = [1,2,3], b = [2,3,4] и c = [3,4,5].

Лабораторная работа № 4

Матрицы и комплексные числа

1. 1 Векторы и матрицы

1.1.1 Векторы

Кратко вспомним, как вводятся векторы. Пусть

a =(-1,2,4) и b=(1.5,2,-1).

Присвоим эти векторные значения переменным a и Ь, вводя либо

» a = [-1 2 4]

» b = [1.5 2 -1]

либо

» a = [-1,2,4]

» b = [1.5,2,-1]

Таким образом, могут быть использованы либо пробелы, либо запятые.

Один способ определения *внутреннего* или *скалярного произведения*, двух векторов a • Ь, был указан ранее (см. лаб. работу № 3). Здесь мы приведем другой способ, использующий идею *поэлементного умножения*. Набрав

» c=a.*Ь

где перед знаком умножения * стоит *точка*, перемножим векторы a и b поэлементно и получим вектор c = [—1.5,4, -4]. Тогда скалярное произведение получается суммированием компонент вектора c:

» sum(c)

что дает a • b = —1.5. Аналогично

» sqrt(sum(a.*a)

дает длину **a**. В действительности, команда MATLAB'a norm сразу найдет длину (норму) вектора. Чтобы найти угол *в* между a и b, мы можем воспользоваться формулой = arccos

a • b/(|a| |b|). В MATLAB'e arccos обозначается через acos, так что вычисления, проводимые таким способом, могут быть записаны как

» theta = acos(sum(a.*b) / sqrt(sum(a.*a)*sum(b.*b)))

что дает приблизительно = 1.693 радиан.

1.1.2. Матрицы

Матрица

$$\mathbf{A} = \begin{pmatrix} -1 & 1 & 2 \\ 3 & -1 & 1 \\ -1 & 3 & 4 \end{pmatrix}$$

вводится для вычислений в MATLAB'e как

» A=[-1 1 2;3 -1 1;-1 3 4]

со строками-векторами, разделенными точками с запятыми, и. как

» A = [-1 1 2
3 - 1 1
-1 3 4]

с различными строками, разделенными <Enter>, и потому появившимися на разных строках экрана. Приглашение MATLAB'a » не появится, пока вы не закончите ввод матрицы закрывающей скобкой]. Если случайно вы написали в одной из строк больше элементов, чем в другой, то получите сообщение об ошибке. Заметьте, что при желании вы можете разделять элементы в строке запятыми:

» A=[-1,1,2;3,-1,1;-1,3,4]

Если вы найдете избыточным число пробелов между числами в строках выдач MATLAB'a, то можете избавиться от лишних с помощью команды

» format compact

На каком-то этапе загляните в help format, чтобы узнать, как можно управлять видом выходных данных MATLAB'a.

Система уравнений

$$\begin{aligned} -x_1 + x_2 - 2x_3 &= 10, \\ 3x_1 - x_2 + x_3 &= -20, \\ -x_1 + 3x_2 + 4x_3 &= 40 \end{aligned} \qquad (2.1)$$

может быть записана как

$$Ax = b, \qquad (2.2)$$

где, скажем, x есть вектор-столбец $(x_1, x_2, x_3,)^T$, a b — вектор-столбец $(10, -20, 40)^T$ правой части этой системы. В MATLAB'e мы можем записать **b** как транспонированный вектор-строку

» b=[10 -20 40]'

Чтобы решить систему линейных уравнений Ax = Ь, где определитель матрицы *A* не равен нулю, мы можем воспользоваться *обратной матрицей* A^{-1} для *A*, т.е. такой матрицей, что $AA^{-1} = A^{-1}A = I$ (I—единичная матрица размера 3 x 3). Чтобы узнать, равен ли нулю ее определитель, наберите

» det(A)

и получите ответ: 10. Тогда решение **x** получается как

$$x = A^{-1}A\,x = A^{-1}\mathbf{b},$$

что в MATLAB'е примет вид

» x=inv(A)*b

отображая результат в виде вектора-столбца **x** (в нашем случае это $(1,19,-4)^{T}$).

Таким образом, матричное произведение получается с использованием символа умножения *, но матрицы должны иметь согласованные размеры, иначе MATLAB выдаст сообщение об ошибке. Например, b*A не имеет смысла, поскольку число столбцов в b, а именно 1, не равно числу строк в A, а именно 3. Вы можете опробовать матричное умножение с помощью

» C=A*A
» det(C)
» D=A^3
» det(D)

Таким образом, $C = A^2$, $D = A^3$ и, по известному правилу для квадратных матриц одинакового порядка, det(P)det(Q) = *det(PQ)*, их определители равны 100 и 1000, соответственно.

На самом деле, чтобы решить линейную систему, подобную (2.1) или (2.2), не надо начинать с обращения соответствующей матрицы. Надо применить линейные преобразования к расширенной матрице, образованной из *A* и b. MATLAB может выполнить эти вычисления следующим образом[1]). Вы просто набираете

» x = A\b

И здесь матрицы должны иметь согласованные размеры. Проделайте это и проверьте ваш ответ, определив произведение Ax или найдя «невязку» r = b — Ax.

Вы можете также сложить две матрицы *одного и того же размера*. MATLAB не будет выражать недовольство, даже если вы прибавите к матрице скаляр. Действительно, результат

» E=A^2+2*A+1

есть матрица

$$E = A^2 + 2A + \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Только не спутайте 1 с единичной матрицей [2] . Единичную матрицу и матрицу, у которой все элементы 1, можно получить, набрав

» diag([l 1 1])
» ones (3,3)

так что diag порождает диагональную матрицу с заданными диагональными элементами, а ones — матрицу заданных размеров из единиц. Посмотрите, что дает

» diag(ones(l,3))

Обратите внимание на скобки при использовании функции diag. В действительности можно получить единичную матрицу (например, размера 3x3), выполнив

» eye(3)

Как уже отмечалось, можно решать систему Ax = Ь, сформировав соответствующую *расширенную* матрицу

» F = [A b]

[2] Из того, что A*1 снова равняется *A*, можно ошибочно подумать, что в MATLAB'е умножение на 1 —это умножение на единичную матрицу.

На самом деле этот метод применим даже тогда, когда определитель равен нулю или матрица *A* системы уравнений не квадратная.

Процедура заключается в приведении расширенной матрицы к «верхней треугольной форме» специального вида, когда с помощью линейных операций над строками создается матрица, у которой ведущие элементы в строках равны 1, а элементы, расположенные перед ними и ниже их, равны нулю.

Верхняя треугольная форма применяется при исследовании магических квадратов. Наберите

» G=rref(F)

и получите эту верхнюю треугольную форму для *F*. В данном случае это

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 19 \\ 0 & 1 & 1 & -4 \end{bmatrix}$$

Это значит, что исходная система *эквивалентна* системе с расширенной матрицей *G*, но последнее есть просто *x* = 1, *y* = 19, z = - 4, так что остается только прочесть решение.

Не забудьте о круглых скобках при использовании функций MATLAB'a, подобных rref. Даже если вы наберете матрицу непосредственно при вызове функции, все равно поставьте их, например, в

```
» rref([1 2 3; 4 5 6; 7 8 9])
```

Обратите внимание, что в данном случае одна строка (последняя) будет состоять из нулей, указывая на линейную зависимость строк исходной матрицы. Можете проверить это, найдя определитель, который равен нулю.

Интересно выполнить

```
» rrefmovie(F)
```

и тогда вы увидите «живую» демонстрацию процесса преобразований матрицы.

2.1.3. Собственные значения и собственные векторы

Если *A* есть n x n-матрица, — число и x — ненулевой n x 1-вектор (столбец), такие, что

$$Ax = \lambda x,$$

то x называется *собственным вектором* матрицы A, а — соответствующим *собственным значением*..[1)] Собственные значения можно понимать и как такие числа , для которых det(A — I) = 0, где *I* — единичная n x n-матрица.

Для примера выполните

```
» A=diag([1 2 3])
» eig(A)
» P=[1 2 3;4 5 6; 5 7 8]
» det(P)
» B=inv(P)*A*P
» eig(B)
» [Y,D]=eig(B)
```

Собственными значениями матрицы *A* будут просто диагональные элементы 1, 2 и 3. Поскольку матрица *P* — невырожденная *(det(P)* = 3), матрицы *A* и $B = P^{-1}AP$ «подобны» и имеют одинаковые собственные значения. Обратите внимание, что MATLAB может упорядочить их по-разному. В последней командной строке определяются собственные векторы матрицы *B* в таком виде: матрица *Y* имеет столбцами собственные векторы, а в диагональной матрице *D* на *диагонали* находятся их собственные значения, причем собственный вектор, соответствующий первому диагональному элементу в *D*, является первым столбцом в Y, и т.д. В данном случае все собственные значения и собственные векторы являются вещественными. Примером, где это не так, служит матрица *A* из §1.1.2.

1) *Геометрически это означает, что линейное преобразование, соответствующее матрице A, переводит x в другой вектор, направленный вдоль той же линии, проходящей через начало координат, что и x.*

1.2. Комплексные числа

MATLAB имеет символ i (-1), встроенный вместе с правилами комплексной арифметики. В MATLAB'e для -1 зарезервирован также символ j. Наберите

```
» clear
» i
» j
» i*i
```

и посмотрите, что получится.

Предупреждение: при работе с комплексными числами есть опасность, что *MATLAB* не будет воспринимать i как квадратный корень из -1. Это может произойти, если перед этим буква i использовалась как индексная переменная, обычно целая (мы позднее встретим индексирование в циклах for). Чтобы оградить себя от этого, просто выполните

```
» clear i,j
```

перед началом работы с комплексными числами и избегайте использования i для других целей.

Вы уже встречались с комплексными числами, когда находили собственные значения и собственные векторы в § 1.1. MATLAB легко управляется с комплексными числами. Например,

```
» a=1+i; b=2-3i;
» c=a*b;
» d=sqrt(a)
```

дают ответы c = 5 — *i*, *d* = 1.0987 + 0.4551i. Заметьте, что приведено только одно значение квадратного корня. Подобно этому

```
» (-1) ^ (1/2)
» (-2+2i) ^ (1/3)
```

дают соответственно *i* и 1 + *i*. Другие значения квадратного или кубического корней получаются умножением на -1 или на кубические корни из единицы, соответственно.

Модуль, аргумент (- и радиан) и вещественная часть уже заданного комплексного числа a получаются посредством

```
» abs(a)
```

» angle (a)
» real (a)
соответственно. Команда
» imag(a)
дает мнимую часть, но без i. Так для a = 1+i ответ есть 1. Команда
» conj (a)
дает $\bar{a}$ — комплексно сопряженное к a. (Если a = *x* + *iy*, где *x* и *y* — вещественные, то a = *x* — *iy.)* Таким образом, для
» a*conj(a)-abs(a) ^ 2
ответом всегда будет 0 (или очень маленькое число, такое, как 1.7764e-015,т.е. 1.7764×10^{-15}).
Попробуйте также выполнить
» exp(i*pi)
что даст $e^{i\pi}=-1$.

2.3. Динамика населения: матрица Лесли

Свойства матриц и их собственных значений широко используются в экономике, в естественных науках и теории вероятностей, а также в физических науках. Очень простой пример дает следующая дискретная по времени модель возрастной структуры населения страны или какой-либо большой группы населения. Основная идея состоит в том, что надо взять вектор, представляющий возрастное распределение за некоторый год, составить матрицу вероятностей перехода от одного года к следующему и затем матричным умножением спрогнозировать вероятностное возрастное распределение возраста на следующий год. Прогноз на последующие годы получится дальнейшим матричным умножением. Самое трудное — построить модель матрицы вероятностей. Приведем пример.

По данным за некоторый год, подсчитаем количество людей в возрастных диапазонах 0-5, 6-19, 20-59 и 60-69. Более подробное деление связано, конечно, с несколько большим объемом работы. Поэтому сделаем следующие довольно сильные упрощающие предположения.

- *Внутри* одного возрастного диапазона возрастное распределение постоянно, т.е. в каждой годовой группе содержится одинаковое число людей.
- Мы не рассматриваем лиц старше 69 лет!
- Смерти случаются лишь в возрастном диапазоне 60-69 лет с интенсивностью d_A% в год и в диапазоне 0-5 лет с интенсивностью d_I % в год.
- Рождения соответствуют возрастному диапазону родителей в 20-59 лет с интенсивностью b % в год.

Возрастной диапазон	Число людей в диапазоне	Число годовых групп	Умерло	Пережило	Покинуло диапазон	Вошло в диапазон
0–5	n_1	6	$\frac{d_I}{100} n_1$	$\left(1-\frac{d_I}{100}\right) n_1$	$\left(1-\frac{d_I}{100}\right) \frac{n_1}{6}$	$\frac{b}{100} n_3$
6–19	n_2	14	0	n_2	$\frac{n_2}{14}$	$\left(1-\frac{d_I}{100}\right) \frac{n_1}{6}$
20–59	n_3	40	0	n_3	$\frac{n_3}{40}$	$\frac{n_2}{14}$
60–69	n_4	10	$\frac{d_A}{100} n_4$	$\left(1-\frac{d_A}{100}\right) n_4$	$\left(1-\frac{d_A}{100}\right) \frac{n_4}{10}$	$\frac{n_3}{40}$

Таблица 2.1. Структура матрицы Лесли.

В этих предположениях мы хотим сопоставить вектор-столбец

$$N(2)=(n_1(2),n_2(2),n_3(2),n_4(2))^T$$

количества людей в четырех возрастных диапазонах за 2-й год с вектором-столбцом

$$N(1)=(n_1(2),n_2(2),n_3(2),n_4(2))^T$$

количества людей за 1-й год. Помните, что, как и выше, T обозначает транспонирование вектора или матрицы.

Чтобы понять, как получить эту взаимосвязь, рассмотрим табл. 2.1.

Ее столбцы содержат: границы возрастного диапазона; обозначение n_i для текущего числа лиц из этого диапазона; число годовых групп (например, возрасты О, 1, 2, 3, 4, 5 образуют шесть годовых групп); число

умерших в этот год согласно нашим правилам; получающееся отсюда число переживших этот год; те из выживших, кто покидает диапазон (переходящих в следующий или, в случае диапазона 60—69, выпадающих из наших расчетов); число включаемых в диапазон, либо как родившихся, либо как перешедших из предыдущего.

Конечно, каждый покинувший диапазон становится вступающим в следующий диапазон. В соответствии с упрощающими предположениями число составляющих возрастной диапазон в каждый отдельный год остается одним и тем же, так что число «покинувших» есть число «выживших», поделенное на число лет, охваченных временным диапазоном. На практике берутся более узкие возраст ные диапазоны, что делает это предположение более правдоподобным.

Теперь легко видеть, что вектор N(2) количества людей разных возрастных диапазонов за 2-й год связан с соответствующим вектором N(1) за 1-й год линейным соотношением

$$N(2) = LN(1),$$

где L — так называемая *матрица Лесли*

$$L = \begin{pmatrix} \frac{5}{6}\left(1 - \frac{d_1}{100}\right) & 0 & \frac{b}{100} & 0 \\ \frac{1}{6}\left(1 - \frac{d_1}{100}\right) & \frac{13}{14} & 0 & 0 \\ 0 & \frac{1}{14} & \frac{39}{40} & 0 \\ 0 & 0 & \frac{1}{40} & \frac{9}{10}\left(1 - \frac{d_A}{100}\right) \end{pmatrix}.$$

По тем же соображениям

$$N(3) = LN(2) = L^2N(1)$$

и т. д., а в общем случае

$$N(t + 1) = LN(t).$$

Задав данные, скажем, для $t = 0$, и некоторые оценки b, d_i и d_A, можно предсказать возможную возрастную структуру населения на несколько лет вперед. Это очень полезно, если вам приходится планировать программы пенсионного обеспечения, университетские штаты или ежедневные поставки продовольствия.

M-файл leslie.m позволит вам исследовать эти понятия. Наберите

```
» leslie
```

и получите 4 x 4-матрицу L, описывающую эволюцию этой модели населения. Вам будет предложено ввести уровень рождаемости *(b)* в процентах в год, например 2.5, уровень детской смертности (например, $d_i = 1$) и смертности по возрасту (например, $d_A = 10$) и т. д. Теперь задайте начальное распределение населения, скажем,

```
» N = [ 3.6 11.4 29.6 10.6]'
```

которое очень грубо аппроксимирует население Великобритании в 1996 г. в млн. чел.

Теперь можно проделать следующее:

Умножьте N на L 4 раза (N1=L*N, N2=L*N1,...), чтобы предсказать на основе этой модели распределение населения Великобритании в 2000 г. Какие же изменения там произойдут? Заметьте, что вы можете ввести и нечто вроде

```
» L^4
» N50=L^50*N
```

Вычислите

```
» [Y D] = eig(L)
```

чтобы получить собственные векторы и собственные значения матрицы Лесли *L*. Найдите наибольшее собственное значение и соответствующий ему собственный вектор **E**. Можно сравнить вектор населения, скажем, после 50 лет, $L^{50}N$, с этим собственным вектором **E** следующим образом:

```
» N50=L^50*N
» N50./E
```

Вторая команда делит компоненты N50 на соответствующие компоненты E. Результатом должны быть три приблизительно равные числа, показывающие, что *население через 50 лет, грубо говоря, пропорционально собственному вектору, соответствующему наибольшему собственному значению.*

Что произойдет, если *исходное* распределение населения будет пропорционально этому собственному вектору?

Упражнения

2.1 В качестве простого примера действий с комплексными числами рассмотрим следующий пример. (Вспомните сделанное в § 2.2 предупреждение: во избежание ошибок перед началом такой работы следует выполнить «clear i».) Введите какое-либо комплексное число, например,

```
» z=3+4i
```

Теперь выполните

```
» z=(z^2-1)/(2*z)
```

Используя клавишу ↑, повторите последнюю команду несколько раз. В конце концов ответы *сходятся* к i или -i; для начального значения 3 + 4i это будет i.

• Сможете ли вы догадаться, как по начальному *z* определить, к чему сойдутся такие числа: к *i* или к -i? (Не ожидаем, что вы сможете доказать это.)

• Если вы начнете с z = 0, то MATLAB будет выдавать —inf, *NaN*, *NaN*..., что в данном случае не имеет

особого смысла[1]). Но есть и другие начальные значения, такие как z = 2, которые приведут к более осмысленной расходимости. Какие начальные значения *z* приводят к расходимости? (И здесь мы ожидаем лишь эмпирический ответ.)

2.2 Это упражнение сочетает работу над матрицами с редактированием файлов и использованием дневников работы. Создайте следующий M-файл (не забудьте либо оставить пробелы между элементами каждой отдельной строки, либо вставьте там запятые):

```
A=[1 1/2 1/3
  1/2 1/3 1/4
  1/3 1/4 1/5]
b=[1 0 0]'
det(A)
X=inv(A)*b
```

и назовите его каким-либо удобным для ссылок именем, например, ch2q2.m.

Этот M-файл решает уравнения с матрицей *A* коэффициентов и вектором-столбцом b в качестве правой части. Заметьте, что сначала вычисляется определитель матрицы *A*. Даже если det(A) □ O, численное решение может быть очень чувствительным к небольшим неточностям или изменениям в параметрах. Этого следует ожидать, если det(A) мал. Данная система оказывается *плохо обусловленной.*

(a) Создайте копию файла ch2q2.m, скажем, ch2q2a.m и отредактируйте ее так, чтобы решались уравнения, в которых элементы 1/3 матрицы *A* заменены на 0.333.

(b) Сделайте то же самое, заменив 1/3 на 0.33; и пусть это будет M-файл ch2q2b.m.

В MATLAB'e есть специальные переменные inf (бесконечность) и NaN (не число), которые могут иметь и знак минус. Первая переменная возникает при выходе текущего результата за принятую в системе числовую шкалу (это примерно 10^{308}), а вторая —в случаях O/O, inf-inf и в арифметических действиях, когда один из операндов уже есть NaN. В программах вместо NaN можно писать nan.

Запустите ваши M-файлы в MATLAB'e и сохраните результаты в дневнике работы. Для этого наберите (в MATLAB'e) diary ch2q2.txt, а затем ch2q2, чтобы выполнить данный M-файл, затем наберите ch2q2a, чтобы прогнать следующий M-файл, и, наконец, ch2q2b, чтобы запустить третий файл. Когда вы прогоните все файлы, наберите **diary off**. Вернитесь к редактору и вызовите файл ch2q2.txt, чтобы просмотреть его содержимое. Вы увидите, что результаты решения этих близких систем уравнений сильно отличаются. Вы, конечно, можете вставить свои собственные комментарии к результатам и распечатать дневник, чтобы получить «твердую копию».

2.3 Скопируйте M-файл leslie.m и отредактируйте его так, чтобы он был удобен для изучения популяции кошек, описываемой возрастными диапазонами 0-1, 2-5, 6-10 и 11-15 и удовлетворяющей следующим модельным предположениям:

- Кошки старше 15 лет не учитываются.
- Рождения происходят в диапазоне 2-5 (b% в год).
- Уровень смертности котят для диапазона 0-1 равен d_K % в год.
- В диапазоне 2-5 выживают все.
- Уровень смертности для диапазона 6-10 равен d_R % и в год, а для диапазона старшего возраста 11-15 равен d_A % в год.

Пусть имя этого M-файла будет cats. m.

(a) Найдите матрицу для случая **b = 15, d_K = 5, d_R =2 и d_A = 30.** (Наберите, например, diary ch2q3.txt, затем запустите M-файл, набрав cats и введя необходимые данные, и в заключение наберите diary off.)

(b) Возрастет или уменьшится вся популяция за пять лет, если ее исходное распределение равно [20, 20, 20, 20]? Вы можете использовать *sum*, чтобы просуммировать все элементы вектора. Можете снова воспользоваться diary ch2q2q3.txt, чтобы включить результат определения величины популяции за этот период и дополнить его своим комментарием: было ли возрастание или уменьшение популяции. Заметьте, что используя *то же* имя для дневника работ, вы добавляете новый материал к старому. Не забывайте выполнять diary off каждый раз, когда вы временно приостанавливаете запись в этот дневник.

(c) Найдите наибольшее по абсолютной величине собственное значение и соответствующий собственный вектор матрицы Лесли C развития популяции кошек. Заметьте, что в

» V=C(:,1) будет первый столбец V матрицы C. Это очень полезно, если вы хотите в дальнейшем что-то сделать с этим столбцом, например, сравнить его с другим вектором.

(d) Используя исходную популяцию из (Ь), покажите, что вектор популяции (N) после 50 лет приблизительно пропорционален собственному вектору из (c). Повторите все, выбрав исходную популяцию по собственному желанию. Вспомните, что » VI./V2 дает вектор, содержащий отношения элементов VI и V2.