5. Задачи линейной алгебры

Познакомимся с инструментами Octave, предназначенными для работы с векторами и матрицами, а также с возможностями, которые предоставляет пакет при непосредственном решении задач линейной алгебры.

5.1 Ввод и формирование векторов и матриц в *Octave*

Векторы и матрицы в Octave задаются путем ввода их элементов.

Элементы *вектора–строки* отделяют пробелами или запятыми, а всю конструкцию заключают в квадратные скобки:

```
>>> a=[2 -3 5 6 -1 0 7 -9]
a =
2 -3 5 6 -1 0 7 -9
>>> b=[-1,0,1]
b =
-1 0 1
```

Листинг 5.1

Вектор–столбец можно задать, если элементы отделять друг от друга точкой с запятой:

```
>>> c=[-pi;-pi/2;0;pi/2;pi]
c =
-3.14159
-1.57080
0.00000
1.57080
3.14159
```

Листинг 5.2

Обратится к элементу вектора можно, указав имя вектора, а в круглых скобках номер элемента под которым он хранится в этом векторе:

```
>>> a(1)
ans = 2
>>> b(3)
ans = 1
>>> c(5)
ans = 3.1416
```

Листинг 5.3

Ввод элементов матрицы так же осуществляется в квадратных скобках, при этом элементы строки отделяются друг от друга пробелом или запятой, а строки разделяются между собой точкой с запятой:

```
>>> Matr=[0 1 2 3;4 5 6 7]
Matr =
0 1 2 3
4 5 6 7
```

Листинг 5.4

Обратиться к элементу матрицы можно, указав после имени матрицы, в круглых скобках, через запятую, номер строки и номер столбца на пересечении которых элемент расположен:

```
>>> Matr(2,3)
ans = 6
```

```
>>> Matr(1,1)
ans = 0
>>> Matr(1,1)=pi;Matr(2,4)=-pi;
>>> Matr
Matr =
3.1416 1.0000 2.0000 3.0000
4.0000 5.0000 6.0000 -3.1416
```

Листинг 5.5

Матрицы и векторы можно *формировать*, составляя их из ранее заданных матриц и векторов:

```
>>> a=[-3 0 2];b=[3 2 -1];c=[5 -2 0];
>>> %Горизонтальная конкатенация векторов-строк,
>>> %результат вектор-строка
>>> M=[a b c]
M = -3 0 2 3 2 -1 5 -2 0
>>> %Вертикальная конкатенация векторов-строк,
>>> %результат матрица
>>> N=[a;b;c]
N =
  -3   0   2
   3   2  -1
   5  -2   0
>>> %Горизонтальная конкатенация матриц
>>> Matrica=[N N N]
Matrica =
  -3   0   2  -3   0   2  -3   0   2
   3   2  -1   3   2  -1   3   2  -1
   5  -2   0   5  -2   0   5  -2   0
>>> %Вертикальная конкатенация матриц
>>> Tablica=[M;M;M]
Tablica =
  -3   0   2   3   2  -1   5  -2   0
  -3   0   2   3   2  -1   5  -2   0
  -3   0   2   3   2  -1   5  -2   0
```

Листинг 5.6

Важную роль при работе с матрицами играет знак двоеточия «:». Примеры с подробными комментариями приведены в листинге 5.7.

```
>>>Tabl=[-1.2 3.4 0.8;0.9 -0.1 1.1;7.6 -4.5 5.6;9.0 1.3 -8.5]
Tabl =
  -1.20000   3.40000   0.80000
   0.90000  -0.10000   1.10000
   7.60000  -4.50000   5.60000
   9.00000   1.30000  -8.50000
>>> %Выделить из матрицы 3-й столбец
>>> Tabl(:,3)
ans =
   0.80000
   1.10000
   5.60000
  -8.50000
>>> %Выделить из матрицы 1-ю строку
```

```
>>> Tabl(1,:)
ans =   -1.20000   3.40000   0.80000
>>> %Выделить из матрицы подматрицу
>>> Matr=Tabl(2:3,1:2)
Matr =
   0.90000  -0.10000
   7.60000  -4.50000
>>> %Вставить подматрицу в правый нижний угол
>>> %исходной матрицы
>>> Tabl(3:4,2:3)=Matr
Tabl =
  -1.20000   3.40000   0.80000
   0.90000  -0.10000   1.10000
   7.60000   0.90000  -0.10000
   9.00000   7.60000  -4.50000
>>> %Удалить из матрицы 2-й столбец
>>> Tabl(:,2)=[]
Tabl =
  -1.20000   0.80000
   0.90000   1.10000
   7.60000  -0.10000
   9.00000  -4.50000
>>> %Удалить из матрицы 2-ю строку
>>> Tabl(2,:)=[]
Tabl =
  -1.20000   0.80000
   7.60000  -0.10000
   9.00000  -4.50000
>>>%Представить матрицу в виде вектора-столбца
>>> Matr
Matr =
   0.90000  -0.10000
   7.60000  -4.50000
>>> Vector=Matr(:)
Vector =
   0.90000
   7.60000
  -0.10000
  -4.50000
>>> %Выделить из вектора элементы со 1-го по 3-й
>>> V=Vector(1:3)
V =
   0.90000
   7.60000
  -0.10000
>>> %Удалить из массива 2-й элемент
>>> V(2)=[]
V =
   0.90000
  -0.10000
```

Листинг 5.7

5.2 Действия над векторами в *Octave*

Рассмотрим *действия над векторами* предусмотренные в Octave.

Для *сложения векторов* используют знак «+» (листинг 5.8). Операция сложения определена только для векторов одного типа, то есть суммировать можно либо векторы–столбцы, либо векторы–строки одинаковой длины.

```
>>> a=[2 4 6];b=[1 3 5];
>>> c=a+b
c =
    3    7   11
```

Листинг 5.8

Вычитание векторов выполняется с помощью знака «–»:

```
>>> a=[2 4 6];b=[1 3 5];
>>> c=a-b
c =
   1   1   1
```

Листинг 5.9

Знак апострофа «′» применяют для *транспонирования вектора:*

```
>>> a'
ans =
   2
   4
   6
>>> b'
ans =
   1
   3
   5
>>> t=(a+b)'
t =
    3
    7
   11
```

Листинг

Умножение вектора на число осуществляется с помощью знака «*».

```
>>> a=[2 4 6];b=[1 3 5];
>>> z=2*a+0.5*b
z =
    4.5000    9.5000   14.5000
```

Листинг 5.10

Знак деления «/» применяют для того, чтобы *разделить вектор на число*:

```
>>> z=2*a+b/2
z =
    4.5000    9.5000   14.5000
```

Листинг 5.11

Умножение вектора на вектор выполняется так же с помощью знака «*». Перемножать можно только векторы одинакового размера, причем один из них должен быть вектором–столбцом, а второй вектором–строкой. Примеры умножения векторов показаны в листинге 5.12.

```
>>> a=[2 4 6];b=[1 3 5];
>>> %В результате умножения вектора-строки
```

```
>>> % на вектор-столбец получится число
>>> a*b'
ans =  44
>>> %В результате умножения вектора-столбца
>>> %на вектор-строку получится матрица
>>> a'*b
ans =
    2    6   10
    4   12   20
    6   18   30
>>> % Некорректное умножение вектоов
>>> a*b
error: operator *: nonconformant arguments
                                (op1 is 1x3, op2 is 1x3)
>>> a'*b'
error: operator *: nonconformant arguments
                                (op1 is 3x1, op2 is 3x1)
```

Листинг 5.12

Все перечисленные действия над векторами определены в математике и относятся к так называемым *векторным вычислениям*. Но Octave допускает и *поэлементное преобразование векторов*. Существуют операции, которые работают с вектором не как с математическим объектом, а как с обычным одномерным массивом. Например, если к некоторому заданному вектору применить математическую функцию, то результатом будет новый вектор того же размера и структуры, но элементы его будут преобразованы в соответствии с заданной функцией (листинг 5.13).

```
>>> x=[-pi/2,-pi/3,-pi/4,0,pi/4,pi/3,pi/2]
x =
-1.5708 -1.0472 -0.7854 0.0000 0.7854 1.0472 1.5708
>>> y=sin(2*x)+cos(2*x)
y =
-1.0000 -1.36603 -1.0000 1.0000 1.0000 0.36603 -1.0000
>>> y=2*exp(x/5)
y = 1.4608 1.6221 1.7093 2.0000 2.3402 2.4660 2.7382
```

Листинг 5.13

Рассмотрим еще несколько операций поэлементного преобразования вектора. К каждому элементу вектора можно *добавить* (*вычесть*) *число*, используя арифметическую операцию «+» («–»):

```
>>> x=[-pi/2,-pi/3,-pi/4,0,pi/4,pi/3,pi/2];
>>> x-1.2+e/3
ans =
-1.86470 -1.34110 -1.07930 -0.29391 0.49149 0.75329 1.27689
```

Листинг 5.14

Поэлементное умножение векторов выполняется при помощи оператора «.*», результатом такого умножения является вектор, каждый элемент которого равен произведению соответствующих элементов заданных векторов.

```
>>> a=[2 4 6];b=[1 3 5];
>>> a.*b
ans =     2   12   30
>>> b.*a
ans =     2   12   30
```

Листинг 5.15

Поэлементное деление одного вектора на другой осуществляется при помощи следующей конструкции «./». В результате получается вектор, каждый элемент которого – частное от деления соответствующего элемента первого вектора на соответствующий элемент второго. Совокупность знаков «.\» применяют для деления векторов в обратном направлении (поэлементное деление второго вектора на первый). Примеры деления векторов показаны в листинге 5.16.

```
>>> a=[2 4 6];b=[1 3 5];
>>> a./b
ans =
   2.0000   1.3333   1.2000
>>> a.\b
ans =
   0.50000   0.75000   0.83333
```

Листинг 5.16

Поэлементное возведение в степень выполняет оператор «.^», результатом является вектор, каждый элемент которого это соответствующий элемент заданного вектора, возведенный в указанную степень (листинг 5.17).

```
>>> a=[2 4 6];b=[1 3 5];
>>>% Каждый элемент вектора возвести в квадат
>>> a.^2
ans =     4   16   36
>>>% Извлечь корень квадратный из каждого элемента вектора
>>> b.^(1/2)
ans =    1.0000   1.7321   2.2361
>>>% Каждый элемент вектора b возвести в степень a
>>> b.^a
ans =        1      81   15625
>>>% Извлечь корень b-й степени
>>>%из каждого элемента вектора a
>>> a.^(1./b)
ans =
   2.0000   1.5874   1.4310
```

Листинг 5.17

5.3 Действиям над матрицами в *Octave.*

Начнем с операций, которые применимы к матрицам с точки зрения классической математики. Одним из базовых действий над матрицами является *сложение* «+» (*вычитание* «–»). Здесь важно помнить, что суммируемые (вычитаемые) матрицы должны быть одной размерности. Результатом такой операции является матрица:

```
>>> Matr_1=[1 2 3;4 5 6;7 8 9]
Matr_1 =
   1   2   3
   4   5   6
 7   8   9
>>> Matr_2=[0 9 8;7 6 5;4 3 2]
Matr_2 =
   0   9   8
   7   6   5
   4   3   2
>>> Matr_3=Matr_1+Matr_2
```

```
Matr_3 =
    1   11   11
   11   11   11
   11   11   11
>>> Matr_4=Matr_2-Matr_1
Matr_4 =
  -1   7   5
   3   1  -1
  -3  -5  -7
```

Листинг 5.18

Умножать на число «*» можно любую матрицу, результатом так же будет матрица, каждый элемент которой будет помножен на заданное число.

```
>>> Matr_1=[1 2 3;4 5 6;7 8 9];
>>> Matr_5=0.2*Matr_1
Matr_5 =
   0.20000   0.40000   0.60000
   0.80000   1.00000   1.20000
   1.40000   1.60000   1.80000
```

Листинг 5.19

Операция транспонирования «'» меняет в заданной матрице строки на столбцы и так же применима к матрицам любой размерности.

```
>>> Matr_5'
ans =
   0.20000   0.80000   1.40000
   0.40000   1.00000   1.60000
   0.60000   1.20000   1.80000
```

Листинг 5.20

При *умножении матриц* «*» важно помнить, что число столбцов первой перемножаемой матрицы должно быть равно числу строк второй. Примеры умножения матриц показаны в листинге 5.21.

```
>>> Matr_1=[1 2 3;4 5 6;7 8 9];
>>> Matr_2=[0 9 8;7 6 5;4 3 2];
>>> Matr_1*Matr_2
ans =
    26    30    24
    59    84    69
    92   138   114
>>> A=[-3 2;0 1];B=[0 -2;3 -1;0 1];
>>> B*A
ans =
   0  -2
  -9   5
   0   1
>>>% Некорректное умножение матриц
>>> A*B
error: operator *: nonconformant arguments
                                (op1 is 2x2, op2 is 3x2)
```

Листинг 5.21

Возведение матрицы в степень «^» эквивалентно ее умножению на себя указанное число раз. При этом целочисленный показатель степени может быть как

положительным, так и отрицательным. Матрица в степени `-1` называется *обратная к данной*. При возведении матрицы в положительную степень выполняется алгоритм умножения матрицы на себя указанное число раз. Возведение в отрицательную степень означает, что умножается на себя матрица обратная к данной. Примеры возведения в степень можно увидеть в листинге 5.22.

```
>>> Matr_6=[3 2 1;1 0 2;4 1 3];
>>> Matr_6^3
ans =
    92    40    59
    65    29    40
   146    65    92
>>> Matr_6^(-1)
ans =
  -0.40000  -1.00000   0.80000
   1.00000   1.00000  -1.00000
   0.20000   1.00000  -0.40000
>>> Matr_6^(-3)
ans =
   0.544000   1.240000  -0.888000
  -1.120000  -1.200000   1.240000
  -0.072000  -1.120000   0.544000
```

Листинг 5.22

Для *поэлементного преобразования матриц* (листинг 5.23) можно применять операции, описанные ранее, как операции поэлементного преобразования векторов: *добавление* (*вычитание*) *числа к каждому элементу матрицы* «+» («−»), *поэлементное умножение матриц* «`.*`» одинакового размера, *поэлементное деление матриц* одинакового размера (прямое «`./`» и обратное «`.\`»), *поэлементное возведение в степень* «`.^`» и *применение к каждому элементу матрицы математических функций*.

```
>>> M=[3 2 1;1 1 2;4 1 3];
>>> N=[4 -2 -1;9 6 -2;-3 -1 2];
>>> 2*M
ans =
   6   4   2
   2   2   4
   8   2   6
>>> N/3
ans =
   1.33333  -0.66667  -0.33333
   3.00000   2.00000  -0.66667
  -1.00000  -0.33333   0.66667
>>> M.*N
ans =
   12   -4   -1
    9    6   -4
 -12   -1    6
>>> N.*M
ans =
   12   -4   -1
    9    6   -4
  -12   -1    6
```

```
>>> M./N
ans =
   0.75000  -1.00000  -1.00000
   0.11111   0.16667  -1.00000
  -1.33333  -1.00000   1.50000
>>> M.\N
ans =
   1.33333  -1.00000  -1.00000
   9.00000   6.00000  -1.00000
  -0.75000  -1.00000   0.66667
>>> M.^0.2
ans =
   1.2457   1.1487   1.0000
   1.0000   1.0000   1.1487
   1.3195   1.0000   1.2457
>>> N.^M
ans =
   64    4   -1
    9    6    4
   81   -1    8
```

Листинг 5.23

Рассмотрим работу с матрицами на следующем примере.

ПРИМЕР 5.1. Вычислить математическое выражение $(2\mathrm{A}+\frac{1}{3}B^T)^2-AB^{-1}$ для заданных значений A и B.

Решение задачи показано в листинге 5.22.

```
>>> A=[-3 2 0;0 1 2;5 3 1];B=[0 -2 1;3 -1 1;0 1 1];
>>> (2*A+1/3*B')^2-A*B^(-1)
ans =
   32.667  -20.667   20.667
   47.333   26.889   15.667
  -40.333   75.333   31.778
```

Листинг 5.24

Довольно необычное, с точки зрения математики, применение нашлось для операторов «/» и «\». Символ «/» используется для операции называемой *делением матриц слева направо*, соответственно знак «\» применяется для *деления матриц справа налево*. Операция B/A эквивалентна выражению $B\cdot A^{-1}$, ее удобно использовать для решения матричных уравнений вида $X\cdot A=B$:

```
>>> A=[2 -1 2;-1 2 -2;2 -2 5]
A =
   2  -1   2
  -1   2  -2
   2  -2   5
>>> B=[7 0 0;0 1 0;0 0 1]
B =
   7   0   0
   0   1   0
   0   0   1
>>> X=B/A
```

```
X =
   6.00000   1.00000  -2.00000
   0.14286   0.85714   0.28571
  -0.28571   0.28571   0.42857
>>>% Проверка XA-B=0
>>> X*A-B
ans =
  -8.8818e-16   4.4409e-16   6.6613e-16
   0.0000e+00   2.2204e-16  -2.2204e-16
   0.0000e+00   5.5511e-17  -2.2204e-16
```

Листинг 5.25

Соответственно A\B эквивалентно $A^{-1} \cdot B$ и применяется для решения уравнения $A \cdot X = B$:

```
>>> A=[2 -1 2;-1 2 -2;2 -2 5];
>>> B=[7 0 0;0 1 0;0 0 1];
>>> X=A\B
X =
   6.00000   0.14286  -0.28571
   1.00000   0.85714   0.28571
  -2.00000   0.28571   0.42857
>>>% Проверка AX-B=0
>>> A*X-B
ans =
  -8.8818e-16   0.0000e+00   0.0000e+00
   4.4409e-16   2.2204e-16   5.5511e-17
   6.6613e-16  -2.2204e-16  -2.2204e-16
```

Листинг 5.26

Если предположить, что *x* и *b* это векторы, а *A* - матрица, то получим запись системы линейных алгебраических уравнений в матричной форме *Ax=b*. Это значит, что оператор «\» можно применять *для решения линейных систем* (листинг 5.27).

```
>>> A=[1 2;1 1];
>>> b=[7;6];
>>> x=A\b
x =
   5
   1
>>>% Проверка Ax=b
>>> A*x
ans =
   7
   6
```

Листинг 5.27

5.4 Функции для работы с матрицами и векторами

В Octave существуют специальные функции, предназначенные для работы с матрицами и векторами. Эти функции можно разделить на следующие группы:

- функции для работы с векторами;
- функции для работы с матрицами;
- функции, реализующие численные алгоритмы решения задач линейной алгебры.

Рассмотрим наиболее часто используемые функции.

5.4.1 Функции для работы с векторами

- `length(X)` – определяет дину вектора `X`;

```
>>> X=[1 2 3 4 5 6 7 8 9];
>>> n=length(X)
n =  9
>>> Y=[-2;-1;0;1;2]
Y =
  -2
  -1
   0
   1
   2
>>> m=length(Y)
m =  5
```

Листинг 5.28

- `prod(X)` – вычисляет произведение элементов вектора `X`;

```
>>> X=[1 2 3 4 5 6 7 8 9];
>>> prod(X)
ans =  362880
```

Листинг 5.29

- `cumprod(X)` – формирует вектор того же типа и размера, что и `X`, каждый элемент которого рассчитывается по формулам $x_1, x_1 \cdot x_2, x_1 \cdot x_2 \cdot x_3, \ldots, x_1 \cdot x_2 \cdot \ldots \cdot x_n$, то есть i–й элементу вектора `X` умножается на произведение всех предыдущих элементов;

```
>>> X=[1 2 3 4 5 ];
>>> cumprod(X)
ans =
     1     2     6    24   120
```

Листинг 5.30

- `sum(X)` – вычисляет сумму элементов вектора `X`;

```
>>> X=[1 2 3 4 5 6 7 8 9];
>>> sum(X)
ans = 45
```

Листинг 5.31

- `cumsum(X)` – формирует вектор кумулятивной суммы, это вектор того же типа и размера, что и `X`, каждый элемент которого рассчитывается следующим образом $x_1, x_1 + x_2, x_1 + x_2 + x_3, \ldots, x_1 + x_2 + \ldots + x_n$, то есть к i–му элементу вектора `X` прибавляется сумма всех предыдущих элементов;

```
>>> X=[1 2 3 4 5 ];
>>> cumsum(X)
ans =
    1    3    6   10   15
```

Листинг 5.32

- `diff(X)` – формирует вектор, размер которого на единицу меньше чем у вектора `X`,

а каждый элемент представляет собой разность между двумя соседними элементами массива X, то есть $x_2-x_1, x_3-x_4, \ldots, x_n-x_{n-1}$;

```
>>> X=[1 2 3 4 5 6 7 8 9];
>>> diff(X)
ans =
1 1 1 1 1 1 1 1
```

Листинг 5.33

- min(X) – находит минимальный элемент вектора X, вызов в формате [nomX, nom]=min(X) дает возможность определить минимальный элемент nomX и его номер nom в массиве X;

```
>>> X=[-1 2 3 9 -8 7 5];
>>> min(X)
ans = -8
>>> [Xnom,nom]=min(X)
Xnom = -8
nom =  5
```

Листинг 5.34

- max(X) – находит максимальный элемент массива X или при [nomX,nom]=max(X) определяет максимум и его номер;

```
>>> X=[-1 2 3 9 -8 7 5];
>>> max(X)
ans =  9
>>> [Xnom,nom]=max(X)
Xnom =  9
nom =  4
```

Листинг 5.35

- mean(X) – определяет среднее арифметическое массива X;

```
>> X=[-1 2 3 9 -8 7 5];
>>> Sr=mean(X)
Sr =  2.4286
>>> sum(X)/length(X)
ans =  2.4286
```

Листинг 5.36

- dot(x1,x2) – вычисляет скалярное произведение векторов x1 и x2;

```
>>> x1=[2 -3 0 5 1];x2=[0 1 -2 3 -4];
>>> dot(x1,x2)
ans =  8
>>> sum(x1.*x2)
ans =  8
>>> x1=[2;-3;0];x2=[0;1;-2];
>>> dot(x1,x2)
ans = -3
>>> sum(x1.*x2)
ans = -3
```

Листинг 5.37

- cross(x1,x2) – определяет векторное произведение векторов x1 и x2;

```
>>> x1=[2 -3 0];x2=[0 1 -2];
```

```
>>> x=cross(x1,x2)
x =
   6   4   2
>>> x1=[2;-3;0];x2=[0;1;-2];
>>> x=cross(x1,x2)
x =
   6
   4
   2
```

Листинг 5.38

- `sort(X)` – выполняет сортировку массива `X`;

```
>>>% Сортировка по возрастанию
>>> X=[-1 2 3 9 -8 7 5];
>>> sort(X)
ans =
  -8  -1   2   3   5   7   9
>>>% Сортировка по убыванию
>>> -sort(-X)
ans =
   9   7   5   3   2  -1  -8
```

Листинг 5.39

5.4.2 Функции для работы с матрицами

- `eye(n [, m])` – возвращает единичную матрицу (вектор) соответствующей размерности;

```
>>> eye(4)
ans =
Diagonal Matrix
   1   0   0   0
   0   1   0   0
   0   0   1   0
   0   0   0   1
>>> eye(2,4)
ans =
Diagonal Matrix
   1   0   0   0
   0   1   0   0
>>> eye(3,1)
ans =
Diagonal Matrix
   1
   0
   0
>>> eye(1,5)
ans =
Diagonal Matrix
   1   0   0   0   0
```

Листинг 5.40

- `ones((n [, m, p, …])` – формирует матрицу (вектор), состоящую из единиц;

```
>>> ones(2)
ans =
   1   1
   1   1
>>> ones(3,3)
ans =
   1   1   1
   1   1   1
   1   1   1
>>> ones(1,4)
ans =
   1   1   1   1
>>> ones(2,1)
ans =
   1
   1
>>> ones(4,2)
ans =
   1   1
   1   1
   1   1
   1   1
>>> ones(2,3,4)
ans =
ans(:,:,1) =
   1   1   1
   1   1   1
ans(:,:,2) =
   1   1   1
   1   1   1
ans(:,:,3) =
   1   1   1
   1   1   1
ans(:,:,4) =
   1   1   1
   1   1   1
```

Листинг 5.41

- `zeros(n [, m, p, …])` – возвращает нулевую матрицу (вектор) соответствующей размерности;

```
>>> zeros(3)
ans =
   0   0   0
   0   0   0
   0   0   0
>>> zeros(1,1)
ans = 0
>>> zeros(1,2)
ans =
   0   0
>>> zeros(3,2)
```

```
ans =
   0   0
   0   0
   0   0
>>> zeros(4,1)
ans =
   0
   0
   0
   0
>> zeros(2,2,2)
ans =
ans(:,:,1) =
   0   0
   0   0
ans(:,:,2) =
   0   0
   0   0
```

Листинг 5.42

- `diag(X [, k])` – возвращает квадратную матрицу с элементами `X` на главной диагонали или на `k`–й; функция `diag(M [, k])`, где `M` ранее определенная матрица, в качестве результата выдаст вектор столбец, содержащий элементы главной или `k`–ой диагонали матрицы `M`;

```
>>> X=[-1 2 3 9 -8 7 5];
>>> diag(X)
ans =
Diagonal Matrix
  -1   0   0   0   0   0   0
   0   2   0   0   0   0   0
   0   0   3   0   0   0   0
   0   0   0   9   0   0   0
   0   0   0   0  -8   0   0
   0   0   0   0   0   7   0
   0   0   0   0   0   0   5
>>> diag(X,0)
ans =
Diagonal Matrix
  -1   0   0   0   0   0   0
   0   2   0   0   0   0   0
   0   0   3   0   0   0   0
   0   0   0   9   0   0   0
   0   0   0   0  -8   0   0
   0   0   0   0   0   7   0
   0   0   0   0   0   0   5
>>> x=[2;-3; 0]; diag(X,1)
ans =
   0  -1   0   0   0   0   0   0
   0   0   2   0   0   0   0   0
   0   0   0   3   0   0   0   0
   0   0   0   0   9   0   0   0
```

```
    0    0    0    0    0   -8    0    0
    0    0    0    0    0    0    7    0
    0    0    0    0    0    0    0    5
    0    0    0    0    0    0    0    0
>>> diag(x,1)
ans =
    0    2    0    0
    0    0   -3    0
    0    0    0    0
    0    0    0    0
>>> x=[2;-3; 0];
>>> diag(x,1)
ans =
    0    2    0    0
    0    0   -3    0
    0    0    0    0
    0    0    0    0
>>> diag(x,-1)
ans =
    0    0    0    0
    2    0    0    0
    0   -3    0    0
    0    0    0    0
>>> x=[2;-3; 1];
>>> diag(x,1)
ans =
    0    2    0    0
    0    0   -3    0
    0    0    0    1
    0    0    0    0
>>> diag(x,-1)
ans =
    0    0    0    0
    2    0    0    0
    0   -3    0    0
    0    0    1    0
>>> diag(x,2)
ans =
    0    0    2    0    0
    0    0    0   -3    0
    0    0    0    0    1
    0    0    0    0    0
    0    0    0    0    0
>>> diag(x,-2)
ans =
    0    0    0    0    0
    0    0    0    0    0
    2    0    0    0    0
    0   -3    0    0    0
    0    0    1    0    0
>>> M=[1 2 3;4 5 6;7 8 9]
```

```
M =
   1   2   3
   4   5   6
   7   8   9
>>> diag(M)
ans =
   1
   5
   9
>>> diag(M,1)
ans =
   2
   6
>>> diag(M,-1)
ans =
   4
   8
>>> diag(M,2)
ans =  3
>>> diag(M,-2)
ans =  7
```

Листинг 5.43

- `rand([n, m, p, …])` – возвращает матрицу (вектор) с элементами распределенными по равномерному закону, `rand` без аргументов возвращает одно случайно число;

```
>>> rand(2)
ans =
   0.15907   0.80147
   0.90460   0.40293
>>> rand(3,1)
ans =
   0.279005
   0.031504
   0.529279
>>> rand(1,4)
ans =
   0.85038   0.13899   0.50764   0.82887
>>> rand(2,5)
ans =
   0.782173   0.286649   0.563683   0.969862   0.708655
   0.300415   0.545783   0.011614   0.143827   0.644821
>>> rand
ans =  0.99252
>>> rand
ans =  0.42848
```

Листинг 5.44

- `randn([n, m, p…])` – возвращает матрицу (вектор), элементы которой распределенными по нормальному закону, `randn` без аргументов возвращает одно случайно число;

```
>>> randn(2)
ans =
  -1.04321  -1.81309
   1.09223  -0.83071
>>> randn(2,4)
ans =
  -0.222773  -0.540185   0.026355   0.308437
   1.510429   1.360071   0.298315   1.186672
>>> randn(1,3)
ans =
   0.38577  -2.33667  -1.35689
>>> randn(2,1)
ans =
  -0.66235
   0.32907
>>> randn
ans = -1.0607
>>> randn
ans = -0.47825
>>> randn
ans = -0.75891
```

Листинг 5.45

- `linspace(a, b [, n])` – возвращает массив из 100 или из `n` точек равномерно распределенных между значениями `a` и `b`;

```
>>> linspace(a,b,3)
ans =
  -2   0   2
>>> a=-2;b=2;n=5;
>>> linspace(a,b,n)
ans =
  -2  -1   0   1   2
>>> linspace(a,b,3)
ans =
  -2   0   2
>>> linspace(0,50,5)
ans =
    0.00000   12.50000   25.00000   37.50000   50.00000
```

Листинг 5.46

- `logspace(a, b [, n])` – формирует массив из 50 или из n точек равномерно распределенных в логарифмическом масштабе между значениями `10a` и `10b`; функция `logspace(a, pi)` дает равномерное распределение из 50 точек в интервале от `10a` до `pi`;

```
>>> logspace(1,2,5)
ans =
    10.000    17.783    31.623    56.234   100.000
>>> logspace(2,pi)
ans =
 Columns 1 through 7:
100.0000 93.1815 86.8279 80.9075 75.3908 70.2503 65.4602
```

```
 Columns 8 through 14:
60.9968 56.8377 52.9622 49.3510 45.9860 42.8504 39.9287
 Columns 15 through 21:
37.2061 34.6692 32.3053 30.1025 28.0500 26.1374 24.3552
 Columns 22 through 28:
22.6945 21.1471 19.7052 18.3616 17.1096 15.9430 14.8559
 Columns 29 through 35:
13.8429 12.8991 12.0195 11.2000 10.4363 9.7247 9.0616
 Columns 36 through 42:
8.4438 7.8680 7.3315 6.8316 6.3658 5.9318 5.5273
 Columns 43 through 49:
5.1504 4.7992 4.4720 4.1671 3.8829 3.6182 3.3715
 Column 50:
    3.1416
```

Листинг 5.47

- `repmat(M, n [, m])` – формирует матрицу состоящую `n` на `n` или из `n` на `m` копий матрицы `M`, если `M` – скаляр, то формируется матрица, элементы которой равны значению `M`;

```
>>> M=[1 2 3;4 5 6;7 8 9];
>>> repmat(A,2)
ans =
   3  -1   3  -1
   6  -2   6  -2
   3  -1   3  -1
   6  -2   6  -2
>>> M=[1 2 3;4 5 6;7 8 9];
>>> repmat(M,2)
ans =
   1   2   3   1   2   3
   4   5   6   4   5   6
   7   8   9   7   8   9
   1   2   3   1   2   3
   4   5   6   4   5   6
   7   8   9   7   8   9
>>> repmat(M,2,3)
ans =
   1   2   3   1   2   3   1   2   3
   4   5   6   4   5   6   4   5   6
   7   8   9   7   8   9   7   8   9
   1   2   3   1   2   3   1   2   3
   4   5   6   4   5   6   4   5   6
   7   8   9   7   8   9   7   8   9
>>> repmat(M,3,1)
ans =
   1   2   3
   4   5   6
   7   8   9
   1   2   3
   4   5   6
   7   8   9
```

```
   1   2   3
   4   5   6
   7   8   9
>>> repmat(9,3)
ans =
   9   9   9
   9   9   9
   9   9   9
```

Листинг 5.48

- `reshape(M, m, n)` – возвращает матрицу размерностью `m` на `n` сформированную из `M` путем последовательной выборки по столбцам, если `M` не имеет `m` на `n` элементов, то выдается сообщение об ошибке;

```
>>> M=[0 1 2 3;4 5 6 7;8 9 0 1]
M =
   0   1   2   3
   4   5   6   7
   8   9   0   1
>>> reshape(M,3,2)
>>>error: reshape: can't reshape 3x4 array to 3x2 array
>>> reshape(M,3,4)
ans =
   0   1   2   3
   4   5   6   7
   8   9   0   1
>>> reshape(M,4,3)
ans =
   0   5   0
   4   9   3
   8   2   7
   1   6   1
>>> reshape(M,2,6)
ans =
   0   8   5   2   0   7
   4   1   9   6   3   1
>>> reshape(M,6,2)
ans =
   0   2
   4   6
   8   0
   1   3
   5   7
   9   1
>>> reshape(M,1,12)
ans =
   0   4   8   1   5   9   2   6   0   3   7   1
>>> reshape(M,12,1)
ans =
   0
   4
   8
```

```
    1
    5
    9
    2
    6
    0
    3
    7
    1
```

Листинг 5.49

- `cat(n, A, B, [C, …])` – объединяет матрицы `A` и `B` или все входящие матрицы;

```
>>> A=[0 1 2;3 4 5;6 7 8];B=[11 12 13;14 15 16;17 18 19];
>>> cat(2,A,B)
ans =
    0    1    2   11   12   13
    3    4    5   14   15   16
    6    7    8   17   18   19
>>> [A,B]
ans =
    0    1    2   11   12   13
    3    4    5   14   15   16
    6    7    8   17   18   19
>>> cat(1,A,B)
ans =
    0    1    2
    3    4    5
    6    7    8
   11   12   13
   14   15   16
   17   18   19
>>> [A;B]
ans =
    0    1    2
    3    4    5
    6    7    8
   11   12   13
   14   15   16
   17   18   19
>>> x1=[2;-3; 0];x2=[0; 1 ;-2];
>>> cat(2,x1,x2)
ans =
   2   0
  -3   1
   0  -2
>>> cat(1,x1,x2)
ans =
   2
  -3
   0
   0
```

```
   1
  -2
>>> x1=[2 -3  0];x2=[0  1 -2];
>>> cat(2,x1,x2)
ans =
   2  -3   0   0   1  -2
>>> [x1 x2]
ans =
   2  -3   0   0   1  -2
>>> cat(1,x1,x2)
ans =
   2  -3   0
   0   1  -2
>>> [x1 ;x2]
ans =
   2  -3   0
   0   1  -2
```

Листинг 5.50

- `rot90(M [, k])` – осуществляет поворот матрицы `M` на 90 градусов или на величину `90k`, где `k` – целое число;

```
>> A=[1 2 3;4 5 6]
>>> M=[0 1 2 3;4 5 6 7;8 9 0 1]
M =
   0   1   2   3
   4   5   6   7
   8   9   0   1
>>> rot90(M)
ans =
   3   7   1
   2   6   0
   1   5   9
   0   4   8
>>> rot90(M,2)
ans =
   1   0   9   8
   7   6   5   4
   3   2   1   0
>>> rot90(M,3)
ans =
   8   4   0
   9   5   1
   0   6   2
   1   7   3
```

Листинг 5.51

- `tril(M [, k])` – формирует из матрицы `M` нижнюю треугольную матрицу начиная с главной или с `k`–й диагонали;

```
>>> tril(M)
ans =
   0   0   0   0
   4   5   0   0
```

```
   8   9   0   0
   6   5   4   3
>>> tril(M,1)
ans =
   0   1   0   0
   4   5   6   0
   8   9   0   1
   6   5   4   3
>>> tril(M,-1)
ans =
   0   0   0   0
   4   0   0   0
   8   9   0   0
   6   5   4   0
>>> tril(M,2)
ans =
   0   1   2   0
   4   5   6   7
   8   9   0   1
   6   5   4   3
>>> tril(M,-2)
ans =
   0   0   0   0
   0   0   0   0
   8   0   0   0
   6   5   0   0
>>> X=[-1 2 3 9 -8 7 5];
>>> tril(X)
ans =
  -1   0   0   0   0   0   0
>>> tril(X')
ans =
  -1
   2
   3
   9
  -8
   7
   5
```

Листинг 5.52

- `triu(M [, k])` – формирует из матрицы `M` верхнюю треугольную матрицу начиная с главной или с `k`–й диагонали;

```
>>> M=[0 1 2 3 ;4 5 6 7;8 9 0 1;6 5 4 3]
M =
   0   1   2   3
   4   5   6   7
   8   9   0   1
   6   5   4   3
>>> triu(M)
ans =
```

```
    0   1   2   3
    0   5   6   7
    0   0   0   1
    0   0   0   3
>>> triu(M,1)
ans =
    0   1   2   3
    0   0   6   7
    0   0   0   1
    0   0   0   0
>>> triu(M,-2)
ans =
    0   1   2   3
    4   5   6   7
    8   9   0   1
    0   5   4   3
>>> triu(X)
ans =
  -1   2   3   9  -8   7   5
>>> triu(X')
ans =
  -1
   0
   0
   0
   0
   0
   0
```

Листинг 5.53

- `size(M)` – определяет число строк и столбцов матрицы `A`, результатом ее работы является вектор [`n`, `m`];

```
>>> M=[0 1 2 3 ;4 5 6 7;8 9 0 1;6 5 4 3];
>>> size(M)
ans =
    4   4
>>> X=[-1 2 3 9 -8 7 5];
>>> size(X)
ans =    1   7
>>> size(X')
ans =    7   1
>>> eye(size(M))
ans =
Diagonal Matrix
    1   0   0   0
    0   1   0   0
    0   0   1   0
    0   0   0   1
>>> zeros(size(X))
ans =    0   0   0   0   0   0   0
```

Листинг 5.54

- `prod(M [,k])` – формирует вектор–строку или вектор–столбец, в зависимости от значения `k`, каждый элемент которой является произведением элементов соответствующего столбца или строки матрицы `M`, если значение параметра `k` в конструкции отсутствует, то по умолчанию вычисляются произведения столбцов матрицы; понятно, что результатом работы функции `prod(prod(A))` будет произведение всех элементов матрицы;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3];
>>> prod(M)
ans =
   24  -25   32   18
>>> prod(M,1)
ans =
   24  -25   32   18
>>> prod(M,2)
ans =
     6
   -40
   -12
  -120
>>> prod(prod(M))
ans = -345600
```

Листинг 5.55

- `cumprod(M [,k])` – отличается от функции `cumprod(X)` тем, что операции описанные для нее применяются либо к строкам либо к столбцам матрицы `M`, в зависимости от значения параметра `k` , по умолчанию накапливание произведения выполняется по столбцам матрицы `M`;

```
>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3];
>>> cumprod(M)
ans =
   -1    1   -2    3
   -4    5    2    6
  -12   -5    8    6
   24  -25   32   18
>>> cumprod(M,1)
ans =
   -1    1   -2    3
   -4    5    2    6
  -12   -5    8    6
   24  -25   32   18
>>> cumprod(M,2)
ans =
    -1    -1     2     6
     4    20   -20   -40
     3    -3   -12   -12
    -2   -10   -40  -120
```

Листинг 5.56

- `sum(M [,k])` – формирует вектор–строку или вектор–столбец, в зависимости от значения `k`, каждый элемент которой является суммой элементов соответствующего столбца или строки матрицы `M`, если значение параметра `k` в

конструкции отсутствует, то по умолчанию вычисляются суммы столбцов матрицы; произведение всех элементов матрицы вычисляет функция `sum(sum(M))`;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3];
>>> sum(M)
ans =
    4   10    5    9
>>> sum(M,1)
ans =
    4   10    5    9
>>> sum(M,2)
ans =
    1
   10
    7
   10
>>> sum(sum(M))
ans =  28
```

Листинг 5.57

- `cumsum(M, [k])` – отличается от функции `cumsum(X)` тем, что операции описанные для нее применяются либо к строкам либо к столбцам матрицы `M`, в зависимости от значения параметра `k`, по умолчанию результатом работы функции является матрица кумулятивных сумм столбцов матрицы `M`;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3];
>>> cumsum(M)
ans =
   -1    1   -2    3
    3    6   -3    5
    6    5    1    6
    4   10    5    9
>>> cumsum(M,1)
ans =
   -1    1   -2    3
    3    6   -3    5
    6    5    1    6
    4   10    5    9
>>> cumsum(M,2)
ans =
   -1    0   -2    1
    4    9    8   10
    3    2    6    7
   -2    3    7   10
```

Листинг 5.58

- `diff(M)` – из матрицы `M` размерностью `n` на `m` формирует матрицу размером `n-1` на `m` элементы которой представляют собой разность между элементами соседних строк `M`;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3]
M =
  -1   1  -2   3
```

```
    4    5   -1    2
    3   -1    4    1
   -2    5    4    3
>>> diff(M)
ans =
    5    4    1   -1
   -1   -6    5   -1
   -5    6    0    2
```

Листинг 5.59

- `min(M)` – формирует вектор–строку каждый элемент которой является наименьшим элементом соответствующего столбца матрицы `M`, определить положение этих элементов в матрице можно, если вызвать функцию в формате `[n, m]=min(M)`, где `n` – это вектор минимальных элементов столбцов матрицы `M`, а `m` – вектор номеров строк матрицы `M`, в которых находятся эти элементы, конструкция `min(min(M))` позволит отыскать минимум среди всех элементов матрицы (листинг 5.60); вызов функции в виде `min(M, [], k)` или `[n, m]=min(M, [], k)` позволяет управлять направлением поиска, в частности можно отыскать минимальные элементы и их положение в строках матрицы `M` (листинг 5.61); и наконец функция `min(A, B)` сформирует матрицу (листинг 5.62) из строк `min(A)` и `min(B)`;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3]
M =
   -1    1   -2    3
    4    5   -1    2
    3   -1    4    1
   -2    5    4    3
>>> min(M)
ans =
   -2   -1   -2    1
>>> [n,m]=min(M)
n =
   -2   -1   -2    1
m =
    4    3    1    3
>>> min(M')
ans =
   -2   -1   -1   -2
>>> [n,m]=min(M')
n =
   -2   -1   -1   -2
m =
    3    3    2    1
>>> min(min(M))
ans = -2
>>> [n,m]=min(min(M))
n = -2
m =  1
```

Листинг 5.60

```
>>> min(M,[],1)
```

```
ans =   -2  -1  -2   1
>>> %То же что и min(M), то есть Формирует вектор-строку,
>>> %каждый элемент которой равен минимальному элементу
>>> %в соответствующем столбце матрицы M
>>> min(M,[],2)
ans =
  -2
  -1
  -1
  -2
>>>%Формирует вектор-столбец, каждый элемент которого,
>>>%равен минимальному элементу в соответствующей
>>>%строке матрицы M
>>> [n,m]=min(M,[],2)
n =
  -2
  -1
  -1
  -2
m =
   3
   3
   2
   1
>>> %и их положение в матрице, то есть номера столбцов
>>> %в которых они находятся
```

Листинг 5.61

```
>>> A=[0 1 2;3 4 5;6 7 8];B=[11 12 13;14 15 16;17 18 19];
>>> min(A,B)
ans =
   0   1   2
   3   4   5
   6   7   8
>>> %Первая строка результирующей матрицы равна минимумам
>>> %столбцов матрицы A, а вторая матрицы B
```

Листинг 5.62

- `max(M)` – формирует вектор–строку каждый элемент которой является наибольшим элементом соответствующего столбца матрицы `M`, действия функций `[n, m]= max(M), max(max(M)), max(M, [], k), [n, m]= max(A, [], k), max(A,B)` понятно из примеров листинга 5.63;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3];
>>> max(M)
ans =
   4   5   4   3
>>> [n,m]=max(M)
n =
   4   5   4   3
m =
   2   2   3   1
>>> max(M')
```

```
ans =    3   5   4   5
>>> [n,m]=max(M')
n =
   3   5   4   5
m =
   4   2   3   2
>>> max(max(M))
ans =  5
>>> [n,m]=max(max(M))
n =  5
m =  2
>>> max(M,[],1)
ans =
   4   5   4   3
>>> max(M,[],2)
ans =
   3
   5
   4
   5
>>> [n,m]=max(M,[],2)
n =
   3
   5
   4
   5
m =
   4
   2
   3
   2
>>> A=[0 1 2;3 4 5;6 7 8];B=[11 12 13;14 15 16;17 18 19];
>>> max(A,B)
ans =
   11   12   13
   14   15   16
   17   18   19
```

Листинг 5.63

- `mean(M, [k])` – формирует вектор–строку или вектор–столбец, в зависимости от значения `k`, каждый элемент которого является средним значением элементов соответствующего столбца или строки матрицы `M`, если значение параметра `k` в конструкции отсутствует, то по умолчанию вычисляются средние значения столбцов матрицы; среднее всех элементов матрицы вычисляет функция `mean(mean(M))`;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3];
>>> mean(M)
ans =
   1.0000   2.5000   1.2500   2.2500
>>> mean(M,1)
ans =
```

```
   1.0000   2.5000   1.2500   2.2500
>>> mean(M,2)
ans =
   0.25000
   2.50000
   1.75000
   2.50000
>>> mean(mean(M))
ans =  1.7500
```

Листинг 5.64

- `sort(A)` – выдает матрицу того же размера, что и `M`, каждый столбец которой упорядочен по возрастанию;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3]
M =
  -1   1  -2   3
   4   5  -1   2
   3  -1   4   1
  -2   5   4   3
>>> sort(M)
ans =
  -2  -1  -2   1
  -1   1  -1   2
   3   5   4   3
   4   5   4   3
>>> sort(M')
ans =
  -2  -1  -1  -2
  -1   2   1   3
   1   4   3   4
   3   5   4   5
>>> -sort(-M)
ans =
   4   5   4   3
   3   5   4   3
  -1   1  -1   2
  -2  -1  -2   1
>>> -sort(-M')
ans =
   3   5   4   5
   1   4   3   4
  -1   2   1   3
  -2  -1  -1  -2
```

Листинг 5.65

- `sqrtm(M)` – относится к так называемым матричным функциям и возвращает матрицу `X`, для которой `X*X=A`;

```
>>> A=[1 0 -3;0 1 2;2 0 -1]
A =
   1   0  -3
   0   1   2
   2   0  -1
```

```
>>> X=sqrtm(A)
X =
   1.53024   0.00000  -1.41861
  -0.35349   1.00000   0.94574
   0.94574   0.00000   0.58450
>>> X*X %Проверка
ans =
   1.00000   0.00000  -3.00000
   0.00000   1.00000   2.00000
   2.00000   0.00000  -1.00000
>>>%Извлечение квадратного корня из каждого элемента
матрицы A
>>> Y=sqrt(A)
Y =
1.00000 + 0.00000i 0.00000 + 0.00000i 0.00000 + 1.73205i
0.00000 + 0.00000i 1.00000 + 0.00000i 1.41421 + 0.00000i
1.41421 + 0.00000i 0.00000 + 0.00000i 0.00000 + 1.00000i
>>> %sqrtm(A) и sqrt(A) дают различные результаты
>>> Y*Y%Матричное умножение
ans =
1.00000 + 2.44949i 0.00000 + 0.00000i -1.73205 + 1.73205i
2.00000 + 0.00000i 1.00000 + 0.00000i 1.41421 + 1.41421i
1.41421 + 1.41421i 0.00000 + 0.00000i -1.00000 + 2.44949i
>>> Y.*Y %Поэлементное умножение
ans =
   1.00000   0.00000  -3.00000
   0.00000   1.00000   2.00000
   2.00000   0.00000  -1.00000
```

Листинг 5.66

- `expm(M)` и `logm(M)` – взаимообратные матричные функции, первая вычисляет матричную экспоненту `eA`, а вторая выполняет логарифмирование по основанию `e`;

```
>>> A=[1 0 -3;0 1 2;2 0 -1];
>>> B=expm(A)
B =
  -0.26543   0.00000  -1.05553
   1.98914   2.71828   0.70369
   0.70369   0.00000  -0.96912
>>> logm(B)
ans =
 1.00000 + 0.00000i 0.00000 + 0.00000i -3.00000 - 0.00000i
-0.00000 - 0.00000i 1.00000 + 0.00000i  2.00000 + 0.00000i
 2.00000 + 0.00000i 0.00000 + 0.00000i -1.00000 - 0.00000i
```

Листинг 5.67

5.4.3 Функции, реализующие численные алгоритмы решения задач линейной алгебры

- `det(M)` – вычисляет определитель квадратной матрицы `M`;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3];
```

```
>>> det(M)
ans =         682
>>> A=[1 0 -3;0 1 2;2 0 -1];
>>> det(A)
ans =           5
```

Листинг 5.68

- `trace(M)` – вычисляет след матрицы `M`, то есть сумму элементов главной диагонали;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3]
M =
          -1           1          -2           3
           4           5          -1           2
           3          -1           4           1
          -2           5           4           3
>>> trace(M)
ans =          11
>>> sum(diag(M))
ans =          11
```

Листинг 5.69

- `norm(M [, p])` – возвращает различные виды норм матрицы `M` в зависимости от `p`, если аргумент `p = 1, 2, inf, fro` не задан, то вычисляется вторая норма матрицы `M`;

```
>>>% Вторая норма
>>> norm(M)
ans =     8.5506
>>> norm(M,2)
ans =     8.5506
>>>% Первая норма
>>> norm(M,1)
ans =          12
>>>% Бесконечная норма
>>> norm(M,inf)
ans =          14
>>>% Евклидова норма
>>> norm(M,'fro')
ans =     11.916
```

Листинг 5.70

- `cond(M [, p])` – возвращает число обусловленности матрицы `M`, основанное на норме `p`;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3];
>>> cond(M)
ans =     3.3324
>>> cond(M,2)
ans =     3.3324
>>> cond(M,1)
ans =     6.4927
>>> cond(M,inf)
ans =     6.7742
>>> cond(M,'fro')
```

```
ans =     5.7154
```

Листинг 5.71

- `rcond(M)` – вычисляет величину обратную значению числа обусловленности матрицы относительно первой нормы, если полученная величина близка к единице, то матрица хорошо обусловлена, если к приближается к нулю, то плохо;

```
>>> M=[-1 1 -2 3 ;4 5 -1 2;3 -1 4 1;-2 5 4 3];
>>> rcond(M)
ans =     0.1738
```

Листинг 5.72

- `inv(M)` – возвращает матрицу обратную к `M`;

```
>>> A=[1 0 -3;0 1 2;2 0 -1];
>>> invA=inv(A)
invA =
        -0.2           0         0.6
         0.8           1        -0.4
        -0.4           0         0.2
>>>%Проверка
>>> A*invA
ans =
           1           0  -5.5511e-17
           0           1           0
           0           0           1
```

Листинг 5.73

- `eig(M)` – возвращает вектор собственных значений матрицы `M`, вызов функции в формате `[Matr, D]= eig(M)` даст матрицу `Matr`, столбцы которой – собственные векторы матрицы `M` и диагональную матрицу `D` , содержащую собственные значения матрицы `M`; функция `eig(A, B)`, где `A` и `B` квадратные матрицы, выдает вектор обобщенных собственных значений.

```
>>> M=[3 -2;-4 1]
M =
           3          -2
          -4           1
>>> eig(M)
ans =
           5
          -1
>>> [Matr,D]=eig(M)
Matr =
     0.70711     0.44721
    -0.70711     0.89443
D =
Diagonal Matrix
           5           0
           0          -1
>>> %Проверка A*M=M*D
>>> M*Matr
ans =
      3.5355    -0.44721
     -3.5355    -0.89443
```

```
>>> Matr*D
ans =
      3.5355    -0.44721
     -3.5355    -0.89443
```

Листинг 5.74

- `poly(M)` – возвращает вектор–строку коэффициентов характеристического полинома матрицы `M`;

```
>>> M=[3 -2;-4 1]
M =
           3          -2
          -4           1
>>> poly(M)
ans =
           1          -4          -5
```

Листинг 5.75

- `rref(M)` – осуществляет приведение матрицы `M` к треугольной форме, используя метод исключения Гаусса;

```
>>> M=[3 -2 1 5;6 -4 2 7;9 -6 3 12]
M =
           3          -2           1           5
           6          -4           2           7
           9          -6           3          12
>>> rref(M)
ans =
           1    -0.66667     0.33333           0
           0           0           0           1
           0           0           0           0
```

Листинг 5.76

- `chol(A)` – возвращает разложение по Холецкому для положительно определенной симметрической матрицы A;

```
>>> M=[10 1 1;2 10 1;2 2 10]
M =
          10           1           1
           2          10           1
           2           2          10
>>> chol(M)
ans =
      3.1623     0.31623     0.31623
           0      3.1464     0.28604
           0           0      3.1334
>>> A=[1 2;1 1];%Матрица не симметрическая
>>> chol(A)
error: chol: matrix not positive definite
>>%Матрица содержит отрицательные элементы
>>> M=[3 -2 1 5;6 -4 2 7;9 -6 3 12];
>>> chol(M)
error: CHOL requires square matrix
```

Листинг 5.77

- `lu(M)` – выполняет LU–разложение, функция `[L, U, P]=lu(M)` возвращает три

матрицы: `L` – нижняя треугольная, `U` – верхняя треугольная и `P` – матрица перестановок, причем $P \cdot A = L \cdot U$; функция `lu(M)` без параметров возвращает одну матрицу, которая в свою очередь, является комбинацией матриц `L` и `U`;

```
>>> M=[3 -2 1; 5 6 -4; 2 7 9];
>>> lu(M)
ans =
           5           6          -4
         0.6        -5.6         3.4
         0.4    -0.82143      13.393
>>> [L,U,P]=lu(M)
L =
           1           0           0
         0.6           1           0
         0.4    -0.82143           1
U =
           5           6          -4
           0        -5.6         3.4
           0           0      13.393
P =
Permutation Matrix
   0   1   0
   1   0   0
   0   0   1
>>> L*U
ans =
           5           6          -4
           3          -2           1
           2           7           9
>>> P*M
ans =
           5           6          -4
           3          -2           1
           2           7           9
>>> lu(M)
ans =
           5           6          -4
         0.6        -5.6         3.4
         0.4    -0.82143      13.393
>>> triu(lu(M))
ans =
           5           6          -4
           0        -5.6         3.4
           0           0      13.393
>>> tril(lu(M))
ans =
           5           0           0
         0.6        -5.6           0
         0.4    -0.82143      13.393
```

Листинг 5.78

- `qr((M)` – выполняет QR–разложение, команда `[Q, R, P]=qr(M)` возвращает три

матрицы: ортогональную матрицу `Q`, верхнюю треугольную матрицу `R` и матрицу перестановок `P`, причем $A \cdot P = Q \cdot P$;

```
>>> M=[3 -2 1; 5 6 -4; 2 7 9];
>>> [Q,R,P]=qr(M)
Q =
     0.10102    -0.27448     0.95627
    -0.40406     0.86703     0.29155
     0.90914     0.41584    0.023324
R =
      9.8995      3.7376     0.10102
           0       8.662      4.3434
           0           0      4.3732
P =
Permutation Matrix
   0   0   1
   0   1   0
   1   0   0
>>> M*P-Q*R
ans =
  -1.1102e-15  4.4409e-16  -4.4409e-16
  -4.4409e-16  -1.7764e-15           0
            0           0  -4.4409e-16
```

Листинг 5.79

- `svd(M)` – возвращает вектор сингулярных чисел матрицы, при использовании в формате `[U, S, V]= svd(M)` выполняет сингулярное разложение матрицы `M`, выдает три матрицы: `U` – сформирована из ортонормированных собственных векторов, отвечающих наибольшим собственным значениям матрицы $M \cdot M^T$, `V` – состоит из ортонормированных собственных векторов матрицы $M \cdot M^T$, `S` – диагональная матрица из сингулярных чисел (неотрицательных значений квадратных корней из собственных значений матрицы $M \cdot M^T$), матрицы удовлетворяют условию $A = U \cdot S \cdot V^T$;

```
>>> M=[3 -2 1; 5 6 -4; 2 7 9];
>>> svd(M)
ans =
11.7553
8.5347
3.7377
>>> [U,S,V]=svd(M)
U =
0.0057541 0.0207345 0.9997685
0.2528901 -0.9673161 0.0186059
0.9674779 0.2527245 -0.0108095
S =
Diagonal Matrix
11.7553 0 0
0 8.5347 0
0 0 3.7377
V =
0.27363 -0.50018 0.82155
```

```
0.70421 -0.47761 -0.52534
0.65515 0.72229 0.22154
```

Листинг 5.80

Рассмотрим некоторые задачи линейной алгебры, которые могут быть решены с помощью описанных выше функций.

5.5 Решение некоторых задач алгебры матриц

Напомним основные определения алгебры матриц. Если $m \times n$ выражений расставлены в прямоугольной таблице из *m* строк и *n* столбцов, то говорят о *матрице* размера $m \times n$:

$$\begin{matrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{matrix}$$

Выражения a_{ij} называют *элементами матрицы*. Элементы a_{ii} (i=1..n), стоящие в таблице на линии, проходящей из левого верхнего угла в правый нижний угол квадрата $n \times n$, образуют *главную диагональ матрицы*.

$$\begin{matrix} a_{11} & a_{12} & \dots & \dots & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{ii} & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & \dots & \dots & a_{mn} \end{matrix}$$

Матрица размером $m \times n (m \neq n)$ называется *прямоугольной*.

$$\begin{matrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{matrix} \qquad \begin{matrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{matrix}$$

В случае если $m = n$, то матрицу называют *квадратной матрицей* порядка *n*.

$$\begin{matrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{matrix}$$

В частности матрица типа $1 \times n$ это *вектор–строка:* $a_{11} \quad a_{12} \quad \dots \quad a_{1n}$

Матрица размером $m \times 1$ является в*ектором–столбцом:*

$$\begin{matrix} a_{11} \\ a_{21} \\ \dots \\ a_{m1} \end{matrix}$$

Число (скаляр) можно рассматривать как матрицу типа 1×1 - a_{11} .

Квадратная матрица $A = \{a_{ij}\}$ размером $n \times n$ называется

- нулевой, если все ее элементы равны нулю:

$$\begin{matrix} 0 & 0 & ... & 0 \\ 0 & 0 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & 0 \end{matrix}$$

- верхней треугольной, если все элементы, расположенные ниже главной диагонали, равны нулю:

$$\begin{matrix} a_{11} & a_{12} & ... & a_{1n} \\ 0 & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ 0 & 0 & ... & a_{nn} \end{matrix}$$

- нижней треугольной, если все элементы, расположенные выше главной диагонали, равны нулю:

$$\begin{matrix} a_{11} & 0 & ... & 0 \\ a_{21} & a_{22} & ... & 0 \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{matrix}$$

- диагональной, если все элементы, кроме элементов главной диагонали, равны нулю:

$$\begin{matrix} a_{11} & 0 & ... & 0 \\ 0 & a_{22} & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & a_{nn} \end{matrix}$$

- единичной, если элементы главной диагонали равны единице, а все остальные нулю:

$$\begin{matrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & 1 \end{matrix}$$

Определителем (детерминантом) матрицы A является число $detA$ или Δ, вычисляемое по правилу:

$$det\ A = \sum (-1)^{\lambda} a_{1i_1} a_{2i_2} ... a_{ni_n} ,$$

где сумма распределена на всевозможные перестановки ($i_1, i_2, ..., i_n$) элементов 1, 2, … , n и, следовательно, содержит $n!$ слагаемых, причем $\lambda = 0$, если перестановка четная, и $\lambda = 1$, если перестановка нечетная.

Квадратная матрица называется *невырожденной*, если ее определитель отличен от нуля $det\ A \neq 0$. В противном случае $det\ A = 0$ матрица называется *вырожденной* или *сингулярной*.

С матрицами можно проводить операции *сравнения*, *сложения* и умножения.

Две матрицы $A = \{a_{ij}\}$ и $B = \{b_{ij}\}$ считаются *равными*, если они одного типа, то есть имеют одинаковое число строк и столбцов, и соответствующие элементы их равны $\{a_{ij}\} = \{b_{ij}\}$.

Суммой двух матриц $A = \{a_{ij}\}$ и $B = \{b_{ij}\}$ одинаково типа называется матрица $C = \{c_{ij}\}$ того же типа, элементы которой равны сумме соответствующих элементов матриц $A = \{a_{ij}\}$ и $B = \{b_{ij}\}$:

$$\{c_{ij}\} = \{a_{ij} + b_{ij}\} .$$

Разность матриц $A=\{a_{ij}\}$ и $B=\{b_{ij}\}$ определяется аналогично:

$$\{c_{ij}\}=\{a_{ij}-b_{ij}\} \quad .$$

Произведением числа $\hbar$ на матрицу $A=\{a_{ij}\}$ (или *произведением матрицы на число*) называется матрица, элементы которой получены умножением всех элементов матрицы $A=\{a_{ij}\}$ на число $\hbar$: $\hbar A=\{\hbar\cdot a_{ij}\}$.

Произведением матриц $A=\{a_{ij}\}$ размерностью $m\times n$ и $B=\{b_{ij}\}$ размерностью $p\times s$ является матрица $C=\{c_{ij}\}$ размерностью $m\times s$, каждый элемент которой можно представить формулой

$$C=\{c_{ij}\}=\{a_{i1}b_{1j}+a_{i2}b_{2j}+...+a_{in}b_{nj}\} \quad , (\ i=1..m,\ j=1..s\ ).$$

Таким образом, произведение матриц $A=\{a_{ij}\}$ и $B=\{b_{ij}\}$ имеет смысл тогда и только тогда, когда количество строк матрицы $A=\{a_{ij}\}$ совпадает с количеством столбцов матрицы $B=\{b_{ij}\}$. Кроме того, произведение двух матриц не обладает переместительным законом, то есть $A\cdot B\neq B\cdot A$. В тех случаях, когда $A\cdot B=B\cdot A$, матрицы $A=\{a_{ij}\}$ и $B=\{b_{ij}\}$ называются *перестановочными*.

Если в матрице $A=\{a_{ij}\}$ размерностью $m\times n$ заменить строки соответствующими столбцами, то получится *транспонированная матрица*

$$A^T=\{a_{ji}\} \quad .$$

В частности, для вектора–строки $a=\{a_1 a_2 a_3 ... a_n\}$ транспонированной матрицей является вектор–столбец:

$$a^T=\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ ... \\ a_n \end{pmatrix}$$

Обратной матрицей по отношению к данной матрице $A=\{a_{ij}\}$ размерностью $n\times n$, называется матрица $A^{-1}=\{A_{ij}\}$ того же типа, которая, будучи умноженной как справа, так и слева на данную матрицу, в результате дает единичную матрицу $E=\{\partial_{ij}\}$:

$$A\dot{A}^{-1}=A^{-1}\dot{A}=E \quad .$$

Нахождение обратной матрицы для данной называется *обращением данной матрицы*. Всякая неособенная матрица имеет обратную матрицу.

Перейдем к конкретным задачам.

ЗАДАЧА 5.1. Для матриц *A*, *B* и *C* проверить выполнение следующих тождеств:

- $(A\cdot B)\cdot C=A\cdot(B\cdot C)$;
- $(A^T+B)\cdot C=A^T\cdot C+B\cdot C$.

На листинге 5.81 видно, что матрицы получившиеся в результате вычисления левой и правой частей первого тождества равны, следовательно, первое предположение истинно. Для исследования второго тождества из левой части равенства вычитаем правую и получаем нулевую матрицу, что так же приводит к выводу об истинности предположения.

```
>>>%Исследование первого тождества
>>> A=[1 -2 0; -3 0 4]
A =
1 -2 0
-3 0 4
>>> B=[3 1;2 0;-1 1]
```

```
B =
3 1
2 0
-1 1
>>> C=[1 2;-1 0]
C =
1 2
-1 0
>>> (A*B)*C
ans =
-2 -2
-14 -26
>>> A*(B*C)
ans =
-2 -2
-14 -26
>>>%Исследование второго тождества
>>> (A'+B)*C-(A'*C+B*C)
ans =
   0   0
   0   0
   0   0
```

Листинг 5.81

ЗАДАЧА 5.2. Проверить является ли матрица симметрической. Квадратная матрица называется *симметрической*, если $A^T = A$.

В листинге 5.82 видно, что в результате вычитания из матрицы A транспонированной матрицы A^T получена нулевая матрица, то есть тождество $A^T = A \Rightarrow A^T - A = 0$ выполнено и заданная матрица симметрическая.

```
>>> A=[1 -0.5 1.5;-0.5 0 2.5;1.5 2.5 -2]
A =
 1.00000 -0.50000  1.50000
-0.50000  0.00000  2.50000
 1.50000  2.50000 -2.00000
>>> A-A'
ans =
0 0 0
0 0 0
0 0 0
```

Листинг 5.82

ЗАДАЧА 5.3. Проверить является ли матрица кососимметрической. Квадратная матрица называется *кососимметрической*, если $A^T = -A$.

Проверив равенство $A^T = -A \Rightarrow A^T + A = 0$ для заданной матрицы (листинг 5.83) убеждаемся в его истинности.

```
>>> A=[0 -0.25 0.75;0.25 0 -1.25;-0.75 1.25 0]
A =
   0.00000  -0.25000   0.75000
   0.25000   0.00000  -1.25000
  -0.75000   1.25000   0.00000
>>> A'+A
ans =
```

```
    0    0    0
    0    0    0
    0    0    0
```

Листинг 5.83

ЗАДАЧА 5.4. Проверить является ли матрица ортогональной. Квадратная матрица называется *ортогональной,* если $|A|=det\ A\neq 0$ и $A^T=A^{-1}$.

Для решения поставленной задачи необходимо вычислить определитель заданной матрицы, и убедиться в том, что он не равен нулю. Затем транспонировать исходную матрицу и найти обратную к ней. Если визуально сложно убедиться в том, что транспонированная матрица равна обратной, можно вычислить их разность. В результате должна получиться нулевая матрица (листинг 5.84).

```
>>> A=[0.5 0.7071 0.5;0.7071 0 -0.7071;0.5 -0.7071 0.5]
A =
   0.50000   0.70710   0.50000
   0.70710   0.00000  -0.70710
   0.50000  -0.70710   0.50000
>>> %Определитель матрицы A отличен от нуля
>>> det(A)
ans = -0.99998
>>> %В результате вычитания из транспонированной матрицы A
>>> обратной к ней матрицы получаем нулевую матрицу.
>> Это значит что A – ортогональная.
an>>> A'-inv(A)
ans =
   0.0000e+00  -1.3562e-05   0.0000e+00
  -1.3562e-05   0.0000e+00   1.3562e-05
   0.0000e+00   1.3562e-05   0.0000e+00
```

Листинг 5.84

ЗАДАЧА 5.5. Задана матрица *A*. Показать, что матрица *B=2A-E*, где *E* – единичная матрица, инволютивна. Квадратная матрица *инволютивной*, если $B^2=E$, где E – единичная матрица.

Решение задачи:

```
>>> A=[6 -15;2 -5];
>>> B=2*A-eye(2);
>>> B^2
ans =
   1   0
   0   1
```

Листинг 5.85

ЗАДАЧА 5.6. Решить матричные уравнения $A\cdot X=B$ и $X\cdot A=B$, выполнить проверку.

Матричное уравнение это уравнение вида $A\cdot X=B$ или $X\cdot A=B$, где X это *неизвестная матрица.* Если умножить матричное уравнение на матрицу обратную к *A*, то оно примет вид:

$$A^{-1}\cdot A\cdot X=A^{-1}\cdot B \quad \text{или} \quad X\cdot A\cdot A^{-1}=B\cdot A^{-1} .$$

Так как $A^{-1}\cdot A=A\cdot A^{-1}=E$, а $E\cdot X=X\cdot E=X$, то неизвестную матрицу X можно вычислить так: $X=A^{-1}\cdot B$ или $X=B\cdot A^{-1}$. Понятно, что матричное уравнение имеет единственное решение если *A* и *B* – квадратные матрицы *n*-го порядка и определитель матрицы *A* не равен нулю. Как решить матричное уравнение в Octave, показано в листинге 5.86.

```
>>> A=[ 2 3;-2  6]
A =
   2   3
  -2   6
>>> B=[2 5;2/3 5/3]
B =
   2.00000   5.00000
   0.66667   1.66667
>>> %Решение матричного уравнения A·X=B
>>> %Первый способ
>>> X=A\B
X =
   0.55556   1.38889
   0.29630   0.74074
>>> %Второй способ
>>> X=inv(A)*B
X =
   0.55556   1.38889
   0.29630   0.74074
>>> %Проверка A·X-B=0
>>> A*X-B
ans =
   0.0000e+00   0.0000e+00
  -3.3307e-16  -6.6613e-16
>>> %Решение матричного уравнения X A=B
>>> %Первый способ
>> X=B/A
X =
   1.222222   0.222222
   0.407407   0.074074
>>> %Второй способ
>>> X=B*inv(A)
X =
   1.222222   0.222222
   0.407407   0.074074
>>> %Проверка X A-B=0
>>> X*A-B
ans =
   0.0000e+00   0.0000e+00
   1.1102e-16   0.0000e+00
```

Листинг 5.86

5.6 Решение систем линейных уравнений

Система *m* уравнений с *n* неизвестными вида

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

называется *системой линейных уравнений*, причем x_j – *неизвестные*, a_{ij} – *коэффициенты при неизвестных*, b_i – *свободные коэффициенты* (i=1..m, j=1..n).

Кроме того, система из *m* линейных уравнений с *n* неизвестными может быть описана при помощи матриц: $A\cdot x=b$, где $x=\{x_j\}$ – *вектор неизвестных*, $A=\{A_{ij}\}$ – *матрица коэффициентов при неизвестных* или *матрица системы*, $b=\{b_i\}$ – *вектор свободных членов системы* или *вектор правых частей* (*i*=1..*m*, *j*=1..*n*).

Матрица $(A|b)$, которая формируется путем приписывания к матрице коэффициентов A столбца свободных членов b , называется *расширенной матрицей системы*.

Если все $b_i=0$, то речь идет об *однородной системе линейных уравнений*, иначе говорят о *неоднородной системе*.

Совокупность всех решений системы $(x_{1,}x_{2,}\ldots,x_n)$, называется *множеством решений* или просто *решением системы*. Две системы уравнений называются *эквивалентными*, если они имеют одинаковое множество решений.

Однородные системы линейных уравнений $A\cdot x=0$ всегда разрешимы, так как последовательность $(x_1=0,x_2=0,\ldots,x_n=0)$ удовлетворяет всем уравнениям системы. Решение $(x_1=0,x_2=0,\ldots,x_n=0)$ называют *тривиальным*. Вопрос о решении однородных систем сводится к вопросу о том, существуют ли кроме тривиального другие, нетривиальные решения.

Система линейных уравнений может не иметь ни одного решения и тогда она называется *несовместной*, например, в системе:

$$x_1+x_2=1$$
$$x_1+x_2=3$$

левые части уравнений совпадают, а правые различны, поэтому никакие значения x_1 и x_2 не могут удовлетворить обоим уравнениям сразу.

Если же система линейных уравнений обладает решением, то она называется *совместной*. Совместная система называется *определенной*, если она обладает одним единственным решением, и *неопределенной*, если решений больше чем одно. Так, система

$$x_1+2\mathrm{x}_2=7$$
$$x_1+x_2=6$$

определена и имеет единственное решение $x_1=5,x_2=1$, а система уравнений

$$3\mathrm{x}_1-x_2=1$$
$$6\mathrm{x}_1-2\mathrm{x}_2=2$$

неопределена, так как имеет бесконечное множество решений вида $x_1=k\,,x_2=3\mathrm{k}-1$, где число k произвольно.

Совокупность всех решений неопределенной системы уравнений называется ее *общим решением*, а какое–то одно конкретное решение – *частным*. Частное решение, полученное из общего при нулевых значениях свободных переменных, называется *базисным*.

При определении совместности систем уравнений важную роль играет понятие ранга матрицы. Пусть дана матрица A размером $n\times m$. Вычеркиванием некоторых строк или столбцов из нее можно получить квадратные матрицы *k*-го порядка, определители которых называются *минорами* порядка k матрицы A . Наивысший порядок не равных нулю миноров матрицы A называют *рангом матрицы* и обозначают $r(A)$. Из определения вытекает, что $r(A)\leqslant min(n\,,m)\,,r(A)=0$, только если матрица нулевая и $r(A)=n$ для невырожденной матрицы *n*-го порядка. При элементарных преобразованиях (перестановка строк матрицы, умножение строк на число отличное от нуля и сложение строк) ранг матрицы не изменяется. Итак, если речь идет об исследовании системы на

совместность, следует помнить, что система n линейных уравнений с m неизвестными:

- несовместна, если $r(A|b)>r(A)$;
- совместна, если $r(A|b)=r(A)$, причем при $r(A|b)=r(A)=m$ имеет *единственное решение*, а при $r(A|b)=r(A)<m$ имеет *бесконечно много решений*.

Существует не мало методов для практического отыскания решений систем линейных уравнений. Эти методы разделяют на *точные* и *приближенные*. Метод относится к классу точных, если с его помощью можно найти решение в результате конечного числа арифметических и логических операций. В этом разделе на конкретных примерах будут рассмотрены только точные методы решения систем.

ЗАДАЧА 5.7. Решить систему линейных уравнений

$$2x_1-x_2+5x_3=0$$
$$3x_1+2x_2-5x_3=1$$
$$x_1+x_2-2x_3=4$$

при помощи правила Крамера.

Правило Крамера заключается в следующем. Если определитель $det\,A$ матрицы системы $A\cdot x=b$ из n уравнений с n неизвестными отличен от нуля то система имеет единственное решение $(x_1, x_2, \ldots, x_n)$, определяемое по формулам Крамера $x_i=\frac{det_i}{det}$, где det_i – определитель матрицы, полученной из матрицы системы A заменой *i*–го столбца столбцом свободных членов b . Если определитель матрицы системы равен нулю, это не означает, что система не имеет решений, возможно, что ее нельзя решить по формулам Крамера.

Итак, для решения поставленной задачи необходимо выполнить следующие действия:

- представить систему в матричном виде, то есть сформировать матрицу системы A и вектор правых частей b ;
- вычислить главный определитель $det\,A$;
- сформировать вспомогательные матрицы для вычисления определителей det_i ;
- вычислить определители det_i ;
- найти решение системы по формуле $x_i=\frac{det_i}{det}$.

Листинге 5.87 содержит решение поставленной задачи.

```
disp('Решение СЛАУ методом Крамера');
disp('Матрица системы:');
A=[2 -1 5;3 2 -5;1 1 -2]
disp('Вектор свободных коэффициентов:');
b=[0;1;4]
disp('Главный определитель:');
D=det(A)
disp('Вспомогательные матрицы:');
A1=A; A1(:,1)=b
A2=A; A2(:,2)=b
A3=A; A3(:,3)=b
disp('Вспомогательные определители:');
d(1)=det(A1);
d(2)=det(A2);
```

```
d(3)=det(A3);
d
disp('Вектор решений СЛАУ Ax=b');
x=d/D
disp('Проверка Ax-b=0');
A*x'-b
%-------------------------------------------------
>>>Решение СЛАУ методом Крамера
Матрица системы:
A =
2 -1 5
3 2 -5
1 1 -2
Вектор свободных коэффициентов:
b =
0
1
4
Главный определитель:
D = 6.0000
Вспомогательные матрицы:
A1 =
0 -1 5
1 2 -5
4 1 -2
A2 =
2 0 5
3 1 -5
1 4 -2
A3 =
2 -1 0
3 2 1
1 1 4
Вспомогательные определители:
d =
-17.000 91.000 25.000
Вектор решений СЛАУ Ax=b
x =
-2.8333 15.1667 4.1667
Проверка Ax-b=0
ans =
-4.4409e-15
3.5527e-15
8.8818e-16
```

Листинг 5.87

Решение СЛАУ по формулам Крамера выглядит достаточно громоздко, поэтому на практике его используют довольно редко.

ЗАДАЧА 5.8. Решить систему линейных уравнений из задачи 5.7 методом обратной матрицы.

Метод обратной матрицы: для системы из n линейных уравнений с n неизвестными $A \cdot x = b$, при условии, что определитель матрицы A не равен нулю,

единственное решение можно представить в виде $x = A^{-1} \cdot b$ (вывод формулы см. в задаче 5.6). Итак, для того, чтобы решить систему линейных уравнений методом обратной матрицы, необходимо выполнить следующие действия:

- сформировать матрицу коэффициентов и вектор свободных членов заданной системы;
- решить систему, представив вектор неизвестных как произведение матрицы обратной к матрице системы и вектора свободных членов (листинг 5.88).

```
disp('Решение СЛАУ методом обратной матрицы');
disp('Матрица системы:');
A=[2 -1 5;3 2 -5;1 1 -2]
disp('Вектор свободных коэффициентов:');
b=[0;1;4]
disp('Вектор решений СЛАУ Ax=b');
x=A^(-1)*b
disp('Вектор решений СЛАУ Ax=b с помощью функции inv(A)');
x=inv(A)*b
disp('Проверка Ax=b');
A*x
%----------------------------------------
>>>Решение СЛАУ методом обратной матрицы
Матрица системы:
A =
   2  -1   5
   3   2  -5
   1   1  -2
Вектор свободных коэффициентов:
b =
   0
   1
   4
Вектор решений СЛАУ Ax=b
x =
   -2.8333
   15.1667
    4.1667
Вектор решений СЛАУ Ax=b с помощью функции inv(A)
x =
   -2.8333
   15.1667
    4.1667
Проверка Ax-b=0
ans =
  -5.3291e-15
   1.0000e+00
   4.0000e+00
```

Листинг 5.88

ЗАДАЧА 5.9. Решить систему линейных уравнений

$$2x_1+x_2-5x_3+x_4=8;$$
$$x_1-3x_2-6x_4=9;$$
$$2x_2-x_3+2x_4=-5;$$
$$x_1+4x_2-7x_3+6x_4=0$$

методом Гаусса.

Решение системы линейных уравнений при помощи *метода Гаусса* основывается на том, что от заданной системы, переходят к эквивалентной системе, которая решается проще, чем исходная система.

Метод Гаусса состоит из двух этапов. *Первый этап* это прямой ход, в результате которого расширенная матрица системы путем элементарных преобразований (перестановка уравнений системы, умножение уравнений на число отличное от нуля и сложение уравнений) приводится к ступенчатому виду:

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right) \Rightarrow \left(\begin{array}{cccc|c} 1 & c_{12} & \dots & c_{1n} & d_1 \\ 0 & 1 & \dots & c_{2n} & d_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & d_n \end{array}\right)$$

На втором этапе (обратный ход) ступенчатую матрицу преобразовывают так, чтобы в первых n столбцах получилась единичная матрица:

$$\left(\begin{array}{cccc|c} 1 & 0 & \dots & 0 & x_1 \\ 0 & 1 & \dots & 0 & x_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & x_n \end{array}\right)$$

Последний, $n+1$ столбец этой матрицы содержит *решение системы линейных уравнений*.

Исходя из выше изложенного, порядок решения задачи в Octave (листинг 5.89) следующий:

• сформировать матрицу коэффициентов A и вектор свободных членов b заданной системы;

• сформировать расширенную матрицу системы, объединив A и b ;

• используя функцию `rref` привести расширенную матрицу к ступенчатому виду;

• найти решение системы, выделив последний столбец матрицы, полученной в предыдущем пункте;

• выполнить вычисление $A\cdot x-b$, если в результате получился нулевой вектор, задача решена верно.

```
disp('Решение СЛАУ методом Гаусса');
disp('Матрица системы:');
A=[2 1 -5 1;1 -3 0 -6;0 2 -1 2;1 4 -7 6]
disp('Вектор свободных коэффициентов:');
b=[8;9;-5;0]
disp('Расширенная матрица системы:');
C=rref([A b])
disp('Размерность матрицы C:');
n=size(C)
disp('Вектор решений СЛАУ Ax=b');
x=C(:,n(2))
disp('Проверка Ax-b');
A*x-b
```

```
%--------------------------------------
>>>Решение СЛАУ методом Гаусса
Матрица системы:
A =
   2   1  -5   1
   1  -3   0  -6
   0   2  -1   2
   1   4  -7   6
Вектор свободных коэффициентов:
b =
   8
   9
  -5
   0
Расширенная матрица системы:
C =
   1.00000   0.00000   0.00000   0.00000   3.00000
   0.00000   1.00000   0.00000   0.00000  -4.00000
   0.00000   0.00000   1.00000   0.00000  -1.00000
   0.00000   0.00000   0.00000   1.00000   1.00000
Размерность матрицы C:
n =
   4   5
Вектор решений СЛАУ Ax=b
x =
   3.00000
  -4.00000
  -1.00000
   1.00000
Проверка Ax-b
ans =
   0.0000e+00
   1.7764e-15
  -8.8818e-16
  -1.6653e-15
```

Листинг 5.89

ЗАДАЧА 5.10. Решить систему линейных уравнений из задачи 5.9 с помощью LU–разложения.

Дадим определение разложения матрицы на множители. Если все определители квадратной матрицы A отличны от нуля, то существуют такие нижняя L и верхняя U треугольные матрицы, что $A = L \cdot U$:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ l_{21} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & \dots & 1 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & u_{nn} \end{pmatrix}$$

Если диагональные элементы одной из матриц ненулевые, то такое разложение единственно.

Метод решения системы линейных уравнений с использованием разложения матрицы коэффициентов на множители называют *LU–разложением* или *LU–факторизацией*.

Если матрица A исходной системы $A \cdot x = b$ разложена в произведение треугольных матриц L и U, то можно записать уравнение: $L \cdot U \cdot x = b$.

Введя вектор вспомогательных переменных $y = (y_1, y_2, \ldots, y_n)^T$, уравнение $L \cdot U \cdot x = b$ можно переписать в виде системы:

$$L \cdot y = b$$
$$U \cdot x = y$$

Таким образом, решение системы $A \cdot x = b$ с квадратной матрицей коэффициентов свелось к последовательному решению двух систем с треугольными матрицами коэффициентов.

Обратим внимание на тот факт, что выполнение приведенных расчетов можно интерпретировать как преобразование данной системы к треугольной. Иными словами LU–разложение это другая схема реализации метода Гаусса.

Исходя из средств, которыми располагает Octave, решение поставленной задачи будет выглядеть так (листинг 6.75):

- сформируем матрицу коэффициентов A и вектор свободных членов b заданной системы;
- воспользовавшись функцией `lu(A)`, получим матрицы L (нижняя треугольная матрица), U (верхняя треугольная матрица) и P (матрица перестановок или иначе, матрица, которая демонстрирует, каким образом были переставлены строки исходной матрицы при разложении на множители L и U);
- по сколку в задаче речь идет о решении системы, то элементы вектора b должны занять места соответствующие строкам матрицы A, для чего необходимо выполнить действие $P \cdot b$;
- решим системы уравнений $L \cdot y = b$ относительно y;
- зная U и y найдем решение x системы $U \cdot x = y$.

```
disp('Решение СЛАУ методом LU-разложения');
disp('Матрица системы:');
A=[2 1 -5 1;1 -3 0 -6;0 2 -1 2;1 4 -7 6]
disp('Вектор свободных коэффициентов:');
b=[8;9;-5;0]
disp('LU-разложение:');
[L,U,P]=lu(A)
Y=rref([L P*b])
n=size(Y)
y=Y(:,n(2))
X=rref([U y])
n=size(X)
x=X(:,n(2))
disp('Проверка Ax-b');
A*x-b
%----------------------------------------
>>>Решение СЛАУ методом LU-разложения
Матрица системы:
A =
   2   1  -5   1
   1  -3   0  -6
   0   2  -1   2
   1   4  -7   6
Вектор свободных коэффициентов:
b =
```

```
    8
    9
   -5
    0
LU-разложение:
L =
   1.00000   0.00000   0.00000   0.00000
   0.50000   1.00000   0.00000   0.00000
   0.50000  -1.00000   1.00000   0.00000
   0.00000  -0.57143  -0.21429   1.00000
U =
   2.00000   1.00000  -5.00000   1.00000
   0.00000  -3.50000   2.50000  -6.50000
   0.00000   0.00000  -2.00000  -1.00000
   0.00000   0.00000   0.00000  -1.92857
P =
Permutation Matrix
   1   0   0   0
   0   1   0   0
   0   0   0   1
   0   0   1   0
Y =
   1.00000   0.00000   0.00000   0.00000   8.00000
   0.00000   1.00000   0.00000   0.00000   5.00000
   0.00000   0.00000   1.00000   0.00000   1.00000
   0.00000   0.00000   0.00000   1.00000  -1.92857
n =
   4   5
y =
   8.0000
   5.0000
   1.0000
  -1.9286
X =
   1.00000   0.00000   0.00000   0.00000   3.00000
   0.00000   1.00000   0.00000   0.00000  -4.00000
   0.00000   0.00000   1.00000   0.00000  -1.00000
   0.00000   0.00000   0.00000   1.00000   1.00000
n =
   4   5
x =
   3.0000
  -4.0000
  -1.0000
   1.0000
Проверка Ax-b
ans =
   0.0000e+00
   0.0000e+00
  -8.8818e-16
   6.6613e-16
```

Листинг 5.90

ЗАДАЧА 5.11. Решить систему линейных уравнений

$$\begin{aligned} 3x_1+x_2-x_3+2x_4&=6;\\ -5x_1+x_2+3x_3-4x_4&=-12;\\ 2x_1+x_3-x_4&=1;\\ x_1-5x_2+3x_3+3x_4&=3 \end{aligned}$$

с помощью QR–разложения.

Квадратную матрицу A можно представить в виде произведения ортогональной матрицы Q и верхней треугольной матрицы R. Использование этого свойства матриц при решении системы линейных уравнений называют *методом QR–разложения*.

Идея решения системы этим методом аналогична той, что была описана в предыдущей задаче:

$$A\cdot x=b\Rightarrow Q\cdot R\cdot x=b\Rightarrow(Q\cdot y=b,\ R\cdot x=y)\ .$$

То есть решение системы уравнений с квадратной матрицей коэффициентов сводится к решению двух систем, матрица коэффициентов первой *ортогональная*, второй – *верхняя треугольная*.

Как решить эту задачу средствами Octave, показано в листинге 5.91.

```
disp('Решение линейной системы с помощью QR-разложения');
A=[3,1,-1,2;-5,1,3,-4;2,0,1,-1;1,-5,3,-3]
b=[6;-12;1;3]
[Q,R]=qr(A)
y=Q'*b
X=rref([R y])
x=X(1:4,5:5)
%---------------------------------------
>>>Решение линейной системы с помощью QR-разложения
A =
   3   1  -1   2
  -5   1   3  -4
   2   0   1  -1
   1  -5   3  -3
b =
    6
  -12
    1
    3
Q =
   0.480384   0.303216   0.358483   0.740797
  -0.800641   0.020214   0.545518   0.246932
   0.320256   0.070750   0.735670  -0.592638
   0.160128  -0.950077   0.180800   0.197546
R =
   6.24500  -1.12090  -2.08167   3.36269
   0.00000   5.07381  -3.02205   3.30505
   0.00000   0.00000   2.55614  -2.74318
   0.00000   0.00000   0.00000   0.49386
y =
   13.2906
   -1.2028
```

```
   -3.1172
    1.4816
X =
   1.00000   0.00000   0.00000   0.00000   1.00000
   0.00000   1.00000   0.00000   0.00000  -1.00000
   0.00000   0.00000   1.00000   0.00000   2.00000
   0.00000   0.00000   0.00000   1.00000   3.00000
x =
   1.00000
  -1.00000
   2.00000
   3.00000
```

Листинг 5.91

ЗАДАЧА 6.12. Исследовать систему на совместность и если возможно решить ее:

а)
$$\begin{cases} x_1+2x_2+5x_3=-9 \\ x_1-x_2+3x_3=2 \\ 3x_1-6x_2-x_3=25 \end{cases}$$

б)
$$\begin{cases} x_1-5x_2-8x_3+x_4=3 \\ 3x_1+x_2-3x_3-5x_4=1 \\ x_1-7x_2+2x_4=-5 \\ 11x_2+20x_3-9x_4=2 \end{cases}$$

в)
$$\begin{cases} 4x_1+x_2-3x_3-x_4=0 \\ 2x_1+3x_2+x_3-5x_4=0 \\ x_1-2x_2-2x_3+3x_4=0 \end{cases}$$

Для решения задачи (листинг 5.92.) введем исходные данные , то есть матрицу коэффициентов системы и вектор правых частей. Затем выполним вычисление рангов матрицы коэффициентов и расширенной матрицы системы.

В случае а) ранги матриц равны и совпадают с количеством неизвестных $r(A|b)=r(A)=3$, значит, система совместна и имеет единственное решение.

Вычисление рангов матрицы системы и расширенной матрицы системы б) показывает, что ранг расширенной матрицы больше ранга матрицы системы $r(A|b)>r(A)$, что означает несовместность системы.

В процессе вычислений для системы в) выясняется, что ранг расширенной матрицы равен рангу матрицы системы $r(A|b)=r(A)=3$, но меньше, чем количество неизвестных системы $r(A|b)=r(A)<4$. Значит, система совместна, но имеет бесконечное множество решений.

```
disp('Исследование системы на совместность');
disp('Введите матрицу системы:');
A=input('A=');
disp('Введите вектор свободных коэффициентов:');
b=input('b=');
disp('Размерность системы:');
[n,m]=size(A)
disp('Ранг матрицы системы:');
r=rank(A)
disp('Ранг расширенной матрицы:');
```

```
R=rank([A b])
if r==R
 disp('Система совместна.');
 if r==m
      disp('Система имеет единственное решение.');
      disp('Решение системы методом обратной матрицы:');
           x=inv(A)*b
      disp('Проверка Ax-b=0:');
           A*x-b
      else
           disp('Система имеет бесконечно много решений.');
      end;
else
       disp('Система не совместна');
end;
%----------------------------------------
%Исследование системы а)
>>>Исследование системы на совместность
Введите матрицу системы:
A= [1 2 5;1 -1 3; 3 -6 -1]
Введите вектор свободных коэффициентов:
b= [-9;2;25]
Размерность системы:
n =  3
m =  3
Ранг матрицы системы:
r =  3
Ранг расширенной матрицы:
R =  3
Система совместна.
Система имеет единственное решение.
Решение системы методом обратной матрицы:
x =
   2.0000
  -3.0000
  -1.0000
Проверка Ax-b=0:
ans =
   0.0000e+00
   8.8818e-16
   3.5527e-15
%----------------------------------------
%Исследование системы б)
>>>Исследование системы на совместность
Введите матрицу системы:
A= [1 -5 -8 1;3 1 -3 -5;1 0 -7 2;0 11 20 -9]
Введите вектор свободных коэффициентов:
b= [3;1;-5;2]
Размерность системы:
n =  4
m =  4
```

```
Ранг матрицы системы:
r =  3
Ранг расширенной матрицы:
R =  4
Система не совместна
%--------------------------------------
%Исследование системы в)
>>>Исследование системы на совместность
Введите матрицу системы:
A= [4 1 -3 -1;2 3 1 -5;1 -2 -2 4]
Введите вектор свободных коэффициентов:
b= [0;0;0]
Размерность системы:
n =  3
m =  4
Ранг матрицы системы:
r =  3
Ранг расширенной матрицы:
R =  3
Система совместна.
Система имеет бесконечное множество решений.
```

Листинг 5.92

5.7 Собственные значения и собственные векторы

Пусть A – матрица размерностью $n \times n$. Любой ненулевой вектор x, принадлежащий некоторому векторному пространству, для которого $A \cdot x = \lambda \cdot x$, где λ некоторое число, называется *собственным вектором матрицы* A, а λ – принадлежащим ему или соответствующим ему *собственным значением матрицы* A.

Уравнение $A \cdot x = \lambda \cdot x$ эквивалентно уравнению $(A - \lambda \cdot E) \cdot x = 0$. Это однородная система линейных уравнений, нетривиальные решения которой являются искомыми *собственными векторами*. Она имеет нетривиальные решения только тогда, когда $r(A - \lambda \cdot E) < n$, то есть, если $det(A - \lambda \cdot E) = 0$.

Многочлен $det(A - \lambda \cdot E)$ называется *характеристическим многочленом матрицы* A, а уравнение $det(A - \lambda \cdot E) = 0$ – *характеристическим уравнением матрицы* А. Если λ_i – собственные значения A, то нетривиальные решения однородной системы линейных уравнений $(A - \lambda \cdot E) = 0$ есть *собственные векторы* A, принадлежащие собственному значению λ_i. Множество решений этой системы уравнений называют *собственным подпространством матрицы* A, принадлежащим собственному значению λ_i, каждый ненулевой вектор собственного подпространства является *собственным вектором матрицы* A.

Иногда требуется найти собственные векторы y и собственные значения $\hbar$, определяемые соотношением $A \cdot y = \hbar \cdot B \cdot y, (y \neq 0)$, где B – невырожденная матрица. Векторы y и числа $\hbar$ обязательно являются собственными векторами и собственными значениями матрицы $B^{-1} \cdot A$. Пусть $A = \{a_{ij}\}$ и $B = \{b_{ij}\}$, причем матрица B является положительно определенной, тогда собственные значения $\hbar$ совпадают с корнями уравнения *n*–й степени $det(A - \hbar \cdot B) = det(a_{IJ} - \hbar \cdot b_{ij}) = 0$. Это уравнение называют *характеристическим уравнением* для обобщенной задачи о собственных значениях. Для каждого корня $\hbar$ кратности m существует ровно m линейно независимых собственных векторов y.

ЗАДАЧА 5.12. Найти собственные значения и собственные векторы матрицы A.
На листинге 5.93 показано решение поставленной задачи.

```
disp('Введите матрицу:');
A=input('A=');
[n,m]=size(A);
disp('Вектор собственных значений матрицы A:');
d=eig(A)
[L, D]=eig(A);
disp('L- Матрица собственных векторов:');
L
disp('D - Диагональная матрица собственных значений:');
D
disp('Проверка:');
for i=1:n
  (A-D(i,i)*eye(n))*L(:,i)
end;
%--------------------------------------
Введите матрицу:
A= [5 2 -1;1 -3 2; 4 5 -3]
Вектор собственных значений матрицы A:
d =
   4.9083e+00
  -2.1495e-16
  -5.9083e+00
L- Матрица собственных векторов:
L =
  -0.796113  -0.049326   0.181303
  -0.241044   0.542590  -0.598803
  -0.555069   0.838548   0.780106
D - Диагональная матрица собственных значений:
D =
Diagonal Matrix
   4.9083e+00            0            0
            0  -2.1495e-16            0
            0            0  -5.9083e+00
Проверка:
ans =
  -2.3657e-16
  -4.3585e-16
   7.4420e-16
ans =
   2.7756e-17
  -4.1633e-16
   4.4409e-16
ans =
   2.0632e-16
   1.5772e-15
   2.6606e-16
```

Листинг 5.93

ЗАДАЧА 5.13. Привести заданную матрицу к диагональному виду.
Задача состоит в том, чтобы для квадратной матрицы A подобрать такую

матрицу C, чтобы матрица $B=C^{-1}\cdot A\cdot C$ имела диагональный вид. Эта задача связана с теорией собственных значений, так как разрешима только в том случае, если матрица C состоит из собственных векторов матрицы A.

На листинге 5.94 показано, как можно решить поставленную задачу.

```
disp('Введите матрицу:');
A=input('A=');
format bank;
[C,D]=eig(A);
disp('Диагональная матрица к матрице A:');
D
disp('Проверка B=D');
B=inv(C)*A*C
%---------------------------------------
Введите матрицу:
A= [2 1 3;1 -2 1;3 2 2]
Диагональная матрица к матрице A:
D =
Diagonal Matrix
  5.41     0     0
     0  -1.00     0
     0     0  -2.41
Проверка B=D
B =
  5.41  -0.00  -0.00
  0.00  -1.00  0.00
  -0.00  -0.00  -2.41
```

Листинг 5.94

ЗАДАЧА 5.14. Найти решение обобщенной задачи о собственных значениях для матриц *A* и *B*.

Обобщенную задачу о собственных значениях (листинг 5.95) решают при помощи функции `eig(A,B)`, которая в качестве результата выдает матрицу обобщенных собственные векторов и диагональную матрицу, содержащую обобщенные собственные значения.

```
disp('Введите матрицу A:');
A=input('A=');
disp('Введите матрицу B:');
B=input('B=');
[X,V]=eig(A,B);
disp('Матрица обобщенных собственных векторов:');
X
disp('Матрица обобщенных собственных значений:');
V
disp('Обобщенные собственные значения:');
v=diag(V)
%---------------------------------------
Введите матрицу A:
A= [1 -3;-3 4]
Введите матрицу B:
B= [1 2;-3 1]
Матрица обобщенных собственных векторов:
X =
```

```
  1.00  0.92
  0.00  1.00
Матрица, содержащая обобщенные собственные значения:
V =
Diagonal Matrix
  1.00     0
     0  -0.71
Обобщенные собственные значения:
v =
  1.00
  -0.71
```

Листинг 5.95

5.8 Норма и число обусловленности матрицы

Матричная норма – это некоторая скалярная числовая характеристика, которую ставят в соответствие матрице. В задачах линейной алгебры используются различные матричные нормы:

- *первая норма* $\|A\|_1$ квадратной матрицы :
$$\|A\|_1 = max \sum_{i=1}^{n} |a_{ij}| \quad ;$$
- *вторая норма* $\|A\|_2$ квадратной матрицы $A=\{a_{ij}\}$:
$$\|A\|_2 = \sqrt{\lambda_{max} \cdot (A \cdot A^T)} \quad ,$$
где $\sqrt{\lambda_{max} \cdot (A \cdot A^T)}$ – максимальное собственное значение матрицы $A=\{a_{ij}\}$;
- *евклидова норма* $\|A\|_e$ квадратной матрицы $A=\{a_{ij}\}$:
$$\|A\|_e = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|} \quad ;$$
- *бесконечная норма* $\|A\|_i$ квадратной матрицы $A=\{a_{ij}\}$:
$$\|A\|_i = max \sum_{j=1}^{n} |a_{ij}| \quad .$$

Число обусловленности матрицы A используется для определения меры чувствительности системы линейных уравнений $A \cdot x = b$ к погрешностям задания вектора b . Чем больше число обусловленности, тем более неустойчив процесс нахождения решения системы. Существует несколько вариантов вычисления числа обусловленности, но все они связаны с нормой матрицы, и равны произведению нормы исходной матрицы на норму обратной:

- *число обусловленности* матрицы, вычисленное в норме $\|A\|_1$:
$$N = \|A\|_1 \cdot \|A^{-1}\|_1 \quad ;$$
- *число обусловленности* матрицы, вычисленное в норме $\|A\|_2$:
$$M = \|A\|_2 \cdot \|A^{-1}\|_2 \quad ;$$
- *число обусловленности* матрицы, вычисленное в норме $\|A\|_i$:
$$P = \|A\|_i \cdot \|A^{-1}\|_i \quad ;$$
- *число обусловленности* матрицы, вычисленное в норме $\|A\|_e$:
$$H = \|A\|_e \cdot \|A^{-1}\|_e \quad .$$

ЗАДАЧА 5.15. Вычислить нормы и числа обусловленности матрицы A.

В листинге 5.96 приведен фрагмент документа, в котором происходит вычисление норм матрицы A с помощью функции `norm` и по соответствующим формулам.

Вычисление чисел обусловленности проведено при помощи функции `cond(A)` и по формулам, отражающим зависимость числа обусловленности от соответствующей нормы матрицы.

```
disp('Введите матрицу:');
A=input('A=');
[n,m]=size(A);
disp('Первая норма:');
n_1=norm(A,1)
N_1=max(sum(abs(A)))
disp('Вторая норма:');
n_2=norm(A,2)
N_2= sqrt(max(eig(A*A')))
disp('Бесконечная норма:');
n_i= norm(A,inf)
N_i=max(sum(abs(A')))
disp('Евклидова норма:');
n_e= norm(A,'fro')
N_e= sqrt(sum(diag(A*A')))
disp('Число обусловленности в первой норме:');
c_1= cond(A,1)
C_1= norm(A,1)*norm(inv(A),1)
disp('Число обусловленности во второй норме:');
c_2=cond(A,2)
C_2=norm(A,2)*norm(inv(A),2)
disp('Число обусловленности в бесконечной норме:');
c_i= cond(A,inf)
C_i= norm(A,inf)*norm(inv(A),inf)
disp('Число обусловленности в евклидовой норме:');
c_e= cond(A,'fro')
C_e= norm(A,'fro')*norm(inv(A),'fro')
%---------------------------------------
Введите матрицу:
A= [5 7 6 5;7 10 8 7;6 8 10 9;5 7 9 10]
Первая норма:
n_1 = 33.00
N_1 = 33.00
Вторая норма:
n_2 = 30.29
N_2 = 30.29
Бесконечная норма:
n_i = 33.00
N_i = 33.00
Евклидова норма:
n_e = 30.55
N_e = 30.55
Число обусловленности в первой норме:
c_1 = 4488.00
C_1 = 4488.00
Число обусловленности во второй норме:
c_2 = 2984.09
C_2 = 2984.09
```

```
Число обусловленности в бесконечной норме:
c_i = 4488.00
C_i = 4488.00
Число обусловленности в евклидовой норме:
c_e = 3009.58
C_e = 3009.58
```

Листинг 5.96

5.9 Задачи линейной алгебры в символьных вычислениях

Основы работы с символьными переменными в Octave описаны в п. 2.7. Рассмотрим работу с матрицами, заданными в символьных переменных и выражениях.

Для *определения символической матрицы* служит функция

```
ex_matrix(число строк, число столбцов, элементы матрицы)
```

Например,

```
>>> symbols
%Определение символьных переменных
>>> a = sym ("a");
>>> b = sym ("b");
>>> c = sym ("c");
>>> d = sym ("d");
%Матрица строка
>>> Matr=ex_matrix(1,3,a,b,c)
Matr =
[[a,b,c]]
%Матрица столбец
>>> Matr=ex_matrix(4,1,a,b,c,d)
Matr =
[[a],[b],[c],[d]]
%Матрица 2 на 2
>>> Matr=ex_matrix(2,2,a,b,c,d)
Matr =
[[a,b],[c,d]]
%Матрица 3 на 3
>>> Matr=ex_matrix(3,3,a,0,b,c,1,1,d,0,2)
Matr =
[[a,0.0,b],[c,1.0,1.0],[d,0.0,2.0]]
```

Листинг 5.97

Над символьными матрицами определены операции *сложения*, *вычитания*, *умножения*.

ЗАДАЧА 5.16. Выполнить действия над матрицами $(A+B)(A-B)$ (листинг 5.98).

```
>>> symbols
%Определение символьных переменных
>>> a = sym ("a");
>>> b = sym ("b");
>>> c = sym ("c");
>>> d = sym ("d");
%Определение матриц
>>> A=ex_matrix(2,2,a,b,c,d)
```

```
A =
[[a,b],[c,d]]
>>> B=ex_matrix(2,2,d,b,c,a)
B =
[[d,b],[c,a]]
>>> C=A+B
C =
[[d+a,2*b],[2*c,d+a]]
>>> D=A-B
D =
[[-d+a,0],[0,d-a]]
>>> C*D
ans =
[[-(d-a)*(d+a),2*(d-a)*b],[-2*(d-a)*c,(d-a)*(d+a)]]
```

Листинг 5.98

К сожалению над символьными матрицами не определены операции вычисления определителя и обратной матрицы.

Для *решения системы линейных алгебраических уравнений* можно воспользоваться функцией `symlsolve`.

ЗАДАЧА 5.17. Решить систему линейных алгебраических уравнений $\begin{Bmatrix} ax+by=c \\ x+y=d \end{Bmatrix}$ относительно переменных *x* и *y* (листинг 5.99).

```
>>> x = sym ("x");
>>> y = sym ("y");
>>> a = sym ("a");
>>> b = sym ("b");
>>> c = sym ("c");
>>> d = sym ("d");
>>> sols = symlsolve({a*x+b*y==c,x+y==d},{x,y})
sols =
(
  [1] =
-(a-b)^(-1)*(d*b-c)
  [2] =
-(a-b)^(-1)*(c-d*a)
)
```

Листинг 5.99

ЗАДАЧА 5.18. Решить систему линейных алгебраических уравнений (листинг 5.100)

$$\begin{matrix} x_1+2x_2+5x_3=-9 \\ x_1-x_2+3x_3=2 \\ 3x_1-6x_2-x_3=25 \end{matrix}.$$

```
>>>sols = symlsolve({x1+2*x2+5*x3==-9,x1-x2+3*x3==2,
                    3*x1-6*x2-x3==25},{x1,x2,x3})
sols =
(  [1] =2.0
   [2] =-3.0
   [3] =-1.0
)
```

Листинг 5.100