3 Массивы и матрицы в Scilab. Решение задач линейной алгебры

Для работы с множеством данных удобно использовать массивы. Например, можно создать массив для хранения числовых или символьных данных. В этом случае, вместо создания переменной, для хранения каждого данного, достаточно создать один массив, где каждому элементу будет присвоен порядковый номер.

Таким образом, *массив* множественный тип данных , состоящий из фиксированного числа элементов. Как и любой другой переменной, массиву должно быть присвоено имя.

Переменную, представляющую собой просто список данных, называют *одномерным массивом* или *вектором.* Для доступа к данным, хранящимся в определенном элементе массива, необходимо указать *имя массива* и *порядковый номер* этого элемента, называемый *индексом.*

Если возникает необходимость хранения данных в виде таблиц, в формате строк и столбцов, то необходимо использовать *двумерные массивы* (матрицы). Для доступа к данным, хранящимся в таком массиве, необходимо указать имя массива и два индекса, первый должен соответствовать номеру строки, а второй номеру столбца в которых хранится необходимый элемент.

Значение *нижней границы индексации* в Scilab равно единице. Индексы могут быть только целыми положительными числами.

3.1 Ввод и формирование массивов и матриц

Самый простой способ задать одномерный массив в Scilab имеет вид:

`[name]=Xn:dX:Xk`[1]

где `name` имя переменной , в которую будет записан сформированный массив , `Xn` значение первого элемента массива, `Xk` значение последнего элемента массива , `dX` шаг , с помощью которого формируется каждый следующий элемент массива, то есть значение второго элемента составит `Xn+dX`, третьего `Xn+ dX+dX` и так далее до `Xk`.

Если параметр dX в конструкции отсутствует, это означает, что по умолчанию он принимает значение равное единице, то есть каждый следующий элемент массива равен значению предыдущего плюс один:

`[name]=Xn:Xk`

Переменную заданную как массив можно использовать в арифметических выражениях и в качестве аргумента математических функций. Результатом работы таких операторов являются массивы:

```
--> Xn=-3.5;dX=1.5;Xk=4.5;
--> X=Xn:dX:Xk
X =
-3.5000 -2.0000 -0.5000 1.0000 2.5000 4.0000
--> Y=sin(X/2)
Y =
-0.9840   -0.8415   -0.2474   0.4794   0.9490   0.9093
--> A=0:5
A =
```

1 Здесь и далее квадратные скобки (если не сказано иначе) означають, что выражение, указанное в них может отсутствовать.

```
0 1 2 3 4 5
--> 0:5
ans =
0     1     2     3     4     5
--> ans/2+%pi
ans =
3.1416   3.6416   4.1416   4.6416   5.1416   5.6416
```

Листинг 3.1

Еще один способ задания векторов и матриц в Scilab это их *поэлементный ввод*.

Так для *определения вектор строки* следует ввести имя массива, а затем после знака присваивания, в квадратных скобках через пробел или запятую перечислить элементы массива:

`[name]=x1 x2 … xn`

или

`[name]=x1, x2, …, xn`

Пример ввода вектора строки :

```
--> V=[1 2 3 4 5]
V =
1 2 3 4 5
--> W=[1.1,2.3,-0.1,5.88]
W =
1.1000 2.3000 -0.1000 5.8800
```

Листинг 3.2

Элементы *вектора столбца вводятся* через точку с запятой:

`[name]=x1; x2; …; xn`

Пример ввода вектора столбца :

```
--> X=[1;2;3]
X =
1
2
3
```

Листинг 3.3

Обратиться к элементу вектора можно, указав имя массива и порядковый номер элемента в круглых скобках:

`name(индекс)`

Например:

```
--> W=[1.1,2.3,-0.1,5.88];
--> W(1)+2*W(3)
ans = 0.9000
```

Листинг 3.4

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:

`[name]=[x11, x12, …, x1n; x21, x22, …, x2n;…; xm1, xm2, …, xmn;`

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

```
name(индекс1, индекс2)
```

Далее приведен пример задания матрицы и обращение к ее элементам:

```
--> A=[1 2 3;4 5 6;7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
--> A(1,2)^A(2,2)/A(3,3)
ans =  3.5556
```

Листинг 3.5

Кроме того, матрицы и векторы можно формировать, составляя их из ранее заданных матриц и векторов:

```
--> v1=[1 2 3]; v2=[4 5 6]; v3=[7 8 9];
--> //Горизонтальная конкатенация векторов-строк:
--> V=[v1 v2 v3]
V = 1 2    3    4    5    6    7    8    9
-->//Вертикальная конкатенация векторов-строк,
-->//результат матрица:
--> V=[v1; v2; v3]
V =
1     2    3
4     5    6
7     8    9
-->//Горизонтальная конкатенация матриц:
--> M=[V V V]
M =
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
-->//Вертикальная конкатенация матриц:
--> M=[V;V]
M =
1     2    3
4     5    6
7     8    9
1     2    3
4     5    6
7     8    9
```

Листинг 3.6

Важную роль при работе с матрицами играет знак двоеточия «:». Указывая его вместо индекса при обращении к массиву, можно иметь доступ к группам его элементов. Например:

```
-->//Пусть задана матрица A
--> A=[5 7 6 5; 7 10 8 7;6 8 10 9;5 7 9 10]
--> //Выделить из матрицы A второй столбец
--> A(:,2)
ans =
```

```
7
10
8
7
--> //Выделить из матрицы A третью строку
--> A(3,:)
ans = 6    8    10   9
--> //Выделить из матрицы A подматрицу M
--> M=A(3:4,2:3)
M =
8     10
7     9
--> //Удалить из матрицы A второй столбец
--> A(:,2)=[]
A =
5     8    10
7     7    9
6     10   9
5     9    10
--> //Удалить из матрицы A третью строку
--> A(3,:)=[]
A =
5     8    10
7     7    9
5     9    10
--> //Представить матрицу M в виде вектора-столбца
--> v=M(:)
v =
8
7
10
9
--> //Выделить из вектора v элементы со второго по четвертый
--> b=v(2:4)
b =
7
10
9
--> //Удалить из массива b второй элемент
--> b(2)=[];
```

Листинг 3.7

3.2 Действия над матрицами

Для работы с матрицами и векторами в Scilab предусмотрены следующие операции:

- \+ сложение;
- − вычитание[2];

2 Операции сложения и вычитания определены для матриц одной размерности или векторов одного типа, то есть суммировать (вычитать) можно либо векторы–столбцы, либо векторы–строки одинаковой длины.

- `'` транспонирование[3];
- `*` матричное умножение[4];
- `*` умножение на число;
- `^` возведение в степень[5];
- `\` левое деление[6];
- `/` правое деление[7];
- `.*` поэлементное умножение матриц;
- `.^` поэлементное возведение в степень;
- `.\` поэлементное левое деление;
- `./` поэлементное правое деление;

Пример действий над матрицами:

```
-->A=[1 2 0;-1 3 1;4 -2 5];
-->B=[-1 0 1;2 1 1;3 -1 -1];
-->//Вычислить (AT+B)2 - 2A(0.5BT-A)
-->(A'+B)^2-2*A*(1/2*B'-A)
 ans  =
    10.    8.     24.
    11.    20.    35.
    63.  - 30.    68.
--> //Решить матричные уравнения A·X=B и X·A=B.
-->A=[3 2;4 3];
-->B=[-1 7;3 5];
-->//Решение матричного уравнения AX=B:
-->X=A\B
 X  =
  - 9.     11.
    13.  - 13.
-->//Решение матричного уравнения XA=B:
-->X=B/A
 X  =
  - 31.    23.
  - 11.    9.
-->//Проверка
```

3 Если в некоторой матрице заменить строки соответствующими столбцами, то получится транспонированная матрица.

4 Операция умножения вектора на вектор определена только для векторов одинакового размера, причем один из них должен быть вектором–столбцом, а второй вектором–строкой. Матричное умножение выполняется по правилу «строка на столбец» и допустимо, если количество строк в одной матрице совпадает с количеством столбцов в другой. Кроме того переместительный закон на произведение матриц не распространяется.

5 Возвести матрицу в n-ю степень, значит умножить ее саму на себя n раз. При этом целочисленный показатель степени может быть как положительным, так и отрицательным. В первом случае выполняется алгоритм умножения матрицы на себя указанное число раз, во втором умножается на себя матрица *обратная к данной.*

6 $(\boldsymbol{A}\backslash\boldsymbol{B}) \Rightarrow (\boldsymbol{A}^{-1}\boldsymbol{B})$, операция может быть применима для решения матричного уравнения вида $\boldsymbol{A}\cdot\boldsymbol{X}=\boldsymbol{B}$, где $\boldsymbol{X}$ неизвестный вектор .

7 $(\boldsymbol{B}/\boldsymbol{A}) \Rightarrow(\boldsymbol{B}\cdot\boldsymbol{A}^{-1})$, используют для решения матричных уравнений вида $\boldsymbol{X}\cdot\boldsymbol{A}=\boldsymbol{B}$.

```
-->X*A-B
 ans  =
    0.    0.
    0.    0.
```

Листинг 3.8

Кроме того, если к некоторому заданному вектору или матрице применить математическую функцию, то результатом будет новый вектор или матрица той же размерности, но элементы будут преобразованы в соответствии с заданной функцией:

```
--> x=[0.1 -2.2 3.14 0 -1];
--> sin(x)
ans =
0.0998     -0.8085   0.0016    0    -0.8415
```

Листинг 3.9

3.3 Специальные матричные функции

Для работы с матрицами и векторами в Scilab существуют специальные функции. Рассмотрим наиболее часто используемые из них.

Функции определения матриц:

- `matrix(A [,n,m])` преобразует матрицу `A` в матрицу другого размера;

```
-->D=[1 2;3 4;5 6];
-->matrix(D,2,3)
ans  =
 1.   5.   4.
 3.   2.   6.
-->matrix(D,3,2)
ans  =
 1.   2.
 3.   4.
 5.   6.
-->matrix(D,1,6)
ans  =
 1.   3.   5.   2.   4.   6.
-->matrix(D,6,1)
ans  =
 1.
 3.
 5.
 2.
 4.
 6.
```

Листинг 3.10

- `ones(m,n)` создает матрицу единиц из `m` строк и `n` столбцов[8];

```
-->ones(1,3) //Формируется вектор-строка
ans  =
```

8 Результатом работы функции ones(n1,n2& ,nn) будет многомерная матрица единиц.

```
 1.   1.   1.
-->ones(2,2) //Формируется квадратная матрица
 ans  =
 1.   1.
 1.   1.
-->m=3; n=2;
-->X=ones(m,n) //Формируется матрица размерностью m на n
 X  =
 1.   1.
 1.   1.
 1.   1.
-->M=[1 2 3;4 5 6]
M  =
 1.   2.   3.
 4.   5.   6.
-->//Формируется матрица Y, состоящая из единиц,
-->//той же размерности, что и матрица M
-->Y=ones(M)
 Y  =
 1.   1.   1.
 1.   1.   1.
```

Листинг 3.11

- `zeros(m,n)` создает *нулевую матрицу*[9] из `m` строк и `n` столбцов[10];

```
-->zeros(3,2)
 ans  =
 0.   0.
 0.   0.
 0.   0.
-->M=[1 2 3 4 5];
-->Z=zeros(M)
 Z  =
 0.   0.   0.   0.   0.
```

Листинг 3.12

- `eye(m,n)` формирует *единичную матрицу*[11] из `m` строк и `n` столбцов;

```
-->eye(3,3)
 ans  =
 1.   0.   0.
 0.   1.   0.
 0.   0.   1.
-->eye(5,1)
 ans  =
1.
0.
 0.
```

9 В нулевой матрице все элементы равны нулю.
10 Результатом работы функции zeros(n1,n2& ,nn) будет многомерная матрица нулей.
11 В единичной матрице элементы главной диагонали равны единице, а все остальные нулю .

```
 0.
 0.
-->m=3; n=4;
-->E=eye(m,n)
 E  =
 1.   0.   0.   0.
 0.   1.   0.   0.
 0.   0.   1.   0.
-->M=[0 1;2 3];
-->//Формируется единичная матрица E
-->//той же размерности, что и матрица M
-->E=eye(M)
 E  =
 1.   0.
 0.   1.
-->//Функцию можно использовать без параметров eye(),
-->//В этом случае задается матрица с неопределенными
-->//размерами, которые будут определены после суммирования
-->//с другой, определенной ранее, матрицей.
-->M=[1 2;3 4;5 6]; E=eye();
-->A=E+M
 A  =
 2.   2.
 3.   5.
 5.   6.
-->M-E
 ans  =
 0.   2.
 3.   3.
 5.   6.
```

Листинг 3.13

- `rand(n1,n2,…nn[,fl])` формирует многомерную матрицу *случайных чисел*, необязательный параметр `p` это символьная переменная, с помощью которой можно задать тип распределения случайной величины (`'uniform'` равномерное, `'normal'` гауссовское); `rand(m,n)` формирует матрицу `m` на `n` случайных чисел; `rand(M)` формирует матрицу случайных чисел, размер которой совпадает с размером матрицы `M`; результат функции `rand()` случайное скалярное число ;

```
-->rand(2,2)//Матрица 2 на 2 случайных чисел
 ans  =
      0.2113249      0.0002211
      0.7560439      0.3303271
--> R=rand(2,2,2)//Многомерный массив случайных чисел
R(:,:,1) =
0.9355     0.4103
0.9169     0.8936
R(:,:,2) =
0.0579     0.8132
0.3529     0.0099
-->rand()//Случайное число
```

```
ans  =
     0.6653811
```

Листинг 3.14

- `sparse([i1 j1;i2 j2;…;in jn],[n1,n2,…,nn])` формирует разреженную матрицу[12]; для создания матрицы такого типа необходимо указать индексы ее ненулевых элементов `[i1 j1,i2 j2,…,in jn]`, и их значения `[n1,n2,…,nn]`, индексы одного элемента отделяются друг от друга либо пробелом, либо запятой, а пары индексов соответственно точкой с запятой, значения элементов разделяются запятыми; при попытке просмотреть матрицу подобного типа пользователю будет предоставлено сообщение о ее размерности, а так же значения ненулевых элементов и их местоположение в матрице;
- `full(M)` вывод разреженной матрицы `M` в виде таблицы;

```
-->A=sparse([1 3;3 2;3 5],[4,5,6])
 A  =
(    3,    5) sparse matrix
(    1,    3)        4.
(    3,    2)        5.
(    3,    5)        6.
-->full(A)
 ans  =
 0.   0.   4.   0.   0.
 0.   0.   0.   0.   0.
 0.   5.   0.   0.   6.
```

Листинг 3.15

- `hypermat(D[,V])` создание многомерной матрицы с размерностью заданной вектором `D` и значениями элементов, хранящихся в векторе `V` (использование параметра `V` необязательно).

```
-->//Пример создания матрицы M,
-->//состоящей из трех матриц размерностью два на два,
-->//каждый элемент матрицы – член последовательности
-->//целых чисел от 0 до 11.
-->M=hypermat([2 2 3],0:11)
 M  =
(:,:,1)
 0.   2.
 1.   3.
(:,:,2)
 4.   6.
 5.   7.
(:,:,3)
 8.   10.
 9.   11.
```

Листинг 3.16

- `diag(V[,k])` возвращает квадратную матрицу с элементами `V` на главной диагонали[13]

12 Разреженная матрица матрица ,большинство элементов которой нули .
13 В диагональной матрице все элементы нули, кроме элементов главной диагонали.

или на k й ; функция diag(A [, k]), где A ранее определенная матрица, в качестве результата выдаст вектор столбец, содержащий элементы главной или k ой диагонали матрицы A;

```
--> V=[1,2,3];
--> diag(V)//Диагональная матрица, V на главной диагонали
ans = 1     0   0
      0     2   0
      0     0   3
-->//Диагональная матрица,
-->//V на первой диагонали (выше главной)
--> diag(V,1)
ans = 0     1   0    0
      0     0   2    0
      0     0   0    3
      0  0 0    0
-->//Диагональная матрица,
-->//V на первой диагонали (ниже главной)
--> diag(V,-1)
ans = 0  0 0    0
      1     0   0    0
      0  2 0    0
      0     0   3    0
--> A=[-1 2 0 ;2 1 -1 ;2 1 3]
A =
-1    2     0
 2    1    -1
 2    1     3
--> diag(A) //Главная диагональ матрицы A
ans =
    -1
     1
     3
```

Листинг 3.17

- cat(n, A, B, [C, …]) объединяет матрицы A и B или все входящие матрицы, при n=1 по строкам, при n=2 по столбцам; то же что [A; B] или [A, B];

```
--> A=[1 2;3 4]; B=[5 6 ;7 8];
--> cat(2,A,B)//Объединение матриц
ans =
1     2    5    6
3     4    7    8
--> cat(1,A,B) //Объединение матриц
ans =
1     2
3     4
5     6
7     8
```

Листинг 3.18

- `tril(A[,k])` формирует из матрицы `A` нижнюю треугольную матрицу[14] начиная с главной или с `k` й диагонали ;

```
--> A=[1 2 3;4 5 6;7 8 9]
A =
1     2    3
4     5    6
7     8    9
-->//Нижняя треугольная матрица, начиная с главной диагонали
--> tril(A)
ans =
1     0    0
4     5    0
7     8    9
--> tril(A,0)//Тоже что и tril(A)
ans =
1     0    0
4     5    0
7     8    9
--> tril(A,1)//Нижняя треугольная матрица,
--> //начиная с первой диагонали (выше главной)
ans =
1     2    0
4     5    6
7     8    9
--> tril(A,-2) )//Нижняя треугольная матрица,
--> //начиная со второй диагонали (ниже главной)
ans =
0     0    0
0     0    0
7     0    0
```

Листинг 3.19

- `triu(A[,k])` формирует из матрицы `A` верхнюю треугольную матрицу[15] начиная с главной или с `k` й диагонали ;

```
--> A=[1 2 3;4 5 6;7 8 9];
--> triu(A)//Верхняя треугольная матрица
ans =
1     2    3
0     5    6
0     0    9
--> triu(A,2) )//Верхняя треугольная матрица,
--> //начиная со второй диагонали (выше главной)
ans =
0     0    3
0     0    0
```

14 Матрица называется нижней треугольной, если все элементы расположенные выше главной диагонали равны нулю.

15 Матрица называется верхней треугольной, если все элементы расположенные ниже главной диагонали равны нулю.

```
0     0    0
--> triu(A,-1) )//Верхняя треугольная матрица,
--> //начиная с первой диагонали (ниже главной)
ans =
1     2    3
4     5    6
0     8    9
```

Листинг 3.20

- sort(X) выполняет упорядочивание массива X, если X матрица, сортировка выполняется по столбцам;

```
-->b=[2 0 1]; sort(b) //Сортировка по убыванию
 ans  =
    2.    1.    0.
-->-sort(-b) //Сортировка по возрастанию
 ans  =
    0.    1.    2.
-->A=[1 2 0;-1 3 1;4 -2 5];
-->sort(A) //Сортировка матрицы
 ans  =
    5.    2.    0.
    4.    1.  - 1.
    3.    1.  - 2.
```

Листинг 3.21

Функции вычисления различных числовых характеристик матриц:

- size(V[,fl]) определяет размер массива V, если V двумерный массив, то size(V,1) или size(v,'r') определяют число строк матрицы V, а size(V,2) или size(V,'c') число столбцов;

```
-->M=[1 2;3 4;5 6;7 8];
-->[n,m]=size(M)
 m  =
      2.
 n  =
      4.
-->size(M,1)
 ans  =
      4.
-->size(M,2)
 ans  =
      2.
```

Листинг 3.22

- length(X) определяет количество элементов массива X, если X вектор, его длину; если X матрица, вычисляет общее число ее элементов;

```
--> V=[-1 0 3 -2 1 -1 1];//Вектор-строка
--> length(V)//Длина вектора
```

```
ans =
    7
-->[1 2 3;4 5 6];//Матрица
-->length(ans)//Количество элементов матрицы
 ans  =
    6.
```

Листинг 3.23

- `sum(X[,fl])` вычисляет *сумму* элементов массива `X`, имеет необязательный параметр `fl`; если параметр `fl` отсутствует, то функция `sum(X)` возвращает скалярное значение равное сумме элементов массива; если `fl='r'` или `fl=1`, что то же самое, то функция вернет строку равную поэлементной сумме столбцов матрицы `X`; если `fl='c'` или `fl=2`, то результатом работы функции будет вектор-столбец, каждый элемент которого равен сумме элементов строк матрицы `X`; частный случай применения функции `sum` это вычисление *скалярного произведения векторов*[16];

```
-->M=[1 2 3;4 5 6;7 8 9];
-->Y=sum(M) //Сумма элементов матрицы
Y  = 45.
-->S1=sum(M,1) //Сумма элементов матрицы по столбцам
S1  =
 12   15   18
-->S2=sum(M,2) // Сумма элементов матрицы по строкам
S2  =
 6
 15
 24
--> V=[-1 0 3 -2 1 -1 1];
--> sum(V) //Сумма элементов вектора
ans = 1
-->//Частный случай. Вычисление скалярного произведения
--> a=[1 2 3];b=[2 0 1];
--> sum(a.*b)
ans = 5
```

Листинг 3.24

- `prod(X[,fl])` вычисляет *произведение* элементов массива `X`, работает аналогично функции `sum`;

```
-->prod(M)
ans  = 362880.
-->p1=prod(M,1)
p1  =
 28   80   162
-->p2=prod(M,2)
p2  =
 6
 120
 504
```

16 Скалярное произведение вычисляется по формуле $\bar{a}\cdot\bar{b}=a_1 b_1+a_2 b_2+...+a_n b_n$.

```
--> V=[1,2,3];
--> prod(V) //Произведение элементов вектора
ans = 6
```

Листинг 3.25

- `max(M[,fl])` вычисляет *наибольший элемент* в массиве `M`, имеет необязательный параметр `fl;` если параметр `fl` отсутствует, то функция `max(M)` возвращает максимальный элемент массива `M`; если `fl='r'`, то функция вернет строку максимальных элементов столбцов матрицы `M`; если `fl='c'`, то результатом работы функции будет вектор-столбец, каждый элемент которого равен максимальному элементу соответствующих строк матрицы `M`; функция `[x, nom]=max(M[,fl])` вернет значение максимального элемента `x` и его номер в массиве `nom;`

```
-->M=[5 0 3;2 7 1;0 4 9];
-->max(M)
 ans  =
      9.
-->max(M,'r')
 ans  =
    5.    7.    9.
-->max(M,'c')
 ans  =
    5.
    7.
    9.
-->[x,nom]=max(M)
 nom  =
    3.    3.
 x  =
      9.
```

Листинг 3.26

- `min(M[,fl])` вычисляет *наименьший элемент* в массиве `M`, работает аналогично функции `max`;

```
-->A=[5 10 3 2 7 1 25 4 0];
-->[x,nom]=max(A)
 nom  =
    7.
 x  =
    25.
```

Листинг 3.27

- `mean(M[,fl])` вычисляет *среднее значение* массива `M`, если `M` двумерный массив, то `mean(M,1)` или `mean(M,'r')` определяют среднее значение строк матрицы `M`, а `mean(M,2)` или `mean(M,'c')` среднее значение столбцов ;

```
-->mean(M)
 ans  =
    3.4444444
```

```
-->mean(M,1)
 ans  =
    2.3333333    3.6666667    4.3333333
-->mean(M,2)
 ans  =
    2.6666667
    3.3333333
    4.3333333
```

Листинг 3.28

- `median(M[,fl])` вычисляет *медиану*[17] массива M, работает аналогично функции `mean`;

```
-->M=[5 0 3;2 7 1;0 4 9];
-->median(M)
 ans  = 3.
-->median(M,1)
 ans  =
    2.    4.    3.
-->median(M,2)
 ans  =
    3.
    2.
    4.
```

Листинг 3.29

- `det(M)` вычисляет *определитель*[18] квадратной матрицы M;

```
-->M=[1 0 2;3 2 1;0 3 1];
-->det(M)
 ans  = 17.
-->Z=[1 2 2;0 1 3;2 4 4];
-->det(Z)
 ans  = 0.
```

Листинг 3.30

- `rank(M[,tol])` вычисление *ранга матрицы* M[19] с точностью `tol`.

```
-->M=[1 0 2;3 2 1;0 3 1];
-->rank(M)
 ans  = 3.
-->Z=[1 2 2;0 1 3;2 4 4];
-->rank(Z)
 ans  = 2.
```

Листинг 3.31

- `norm(M[,fl])` вычисление *нормы* квадратной матрицы M, тип нормы определяется необязательной строковой переменной `fl`, по умолчанию `fl=2`; функции `norm(M)` и

17 Значение, которое делит массив на две части.

18 Определитель или детерминант матрицы это число, представляющее собой сумму всевозможных перестановок элементов этой матрицы.

19 *Ранг матрицы* максимальное число линейно независимых строк .

norm(M,2) эквивалентны и вычисляют вторую норму матрицы M[20]; первая норма[21] определяется функцией norm(M,1); функции norm(M,'inf') и norm(M,'fro') вычисляют соответственно бесконечную[22] и евклидову нормы[23]; если V вектор , то результатом работы функции norm(V,1) будет сумма модулей всех элементов вектора V; с помощью функции norm(V,2) можно вычислить модуль вектора V[24]; значение norm(V,'inf') равно модулю максимального элемента вектора по модулю;

```
-->M=[1 0 2;3 2 1;0 3 1];
-->norm(M,1)
 ans  = 5.
-->norm(M,2)
 ans  =
    4.5806705
-->norm(M,'inf')
 ans  = 6.
-->norm(M,'fro')
 ans  = 5.3851648
-->X=[5 -3 4 -1 2];
-->norm(X,1)
 ans  = 15.
-->sum(abs(X))//То же, что и norm(X,1)
 ans  = 15.
-->norm(X,2)
 ans  = 7.4161985
-->sqrt(sum(X^2)) //То же, что и norm(X,2)
 ans  = 7.4161985
-->norm(X,'inf')
 ans  =
    5.
-->max(abs(X))//То же, что и norm(X,'inf')
 ans  =
    5.
```

Листинг 3.32

- cond(M) вычисляет *число обусловленности*[25] матрицы M по второй норме;

```
-->A=[5 7 6 5;7 10 8 7;6 8 10 9;5 7 9 10];
-->cond(A)
 ans  =
    2984.0927
```

Листинг 3.33

20 *Вторая норма матрицы* ее наибольшее сингулярное значение .
21 *Первая норма матрицы* наибольшая сумма по столбцам .
22 Бесконечная норма наибольшая сумма по строкам .
23 Евклидова норма - корень из суммы квадратов всех элементов матрицы.
24 Модуль вектора - корень квадратный из суммы квадратов его элементов.
25 Число обусловленности равно произведению нормы исходной матрицы на норму обратной.

Функции, реализующие численные алгоритмы решения задач линейной алгебры:

- `spec(M)` вычисляет *собственные значения* и *собственные векторы*[26] квадратной матрицы `M`.

```
-->M=[3 -2;-4 1]
 M  =
 3.   - 2.
 - 4. 1.
-->spec(M) //Собственные числа матрицы
 ans  =
 - 1.
   5.
//X - собственные векторы,
-->соответствующие собственным значениям из матрицы Y.
-->[X,Y]=spec(M)
 Y  =
! - 1.    0  !
!   0     5. !
 X  =
!   0.4472136  - 0.7071068 !
!   0.8944272    0.7071068 !
```

Листинг 3.34

- `inv(A)` вычисляет *матрицу обратную*[27] к `A`[28];

```
-->//Пример вычисления обратной матрицы.
-->A=[1 2 3 5;0 1 3 2;4 2 1 1;2 3 0 1];
-->inv(A)
 ans  =
!   0.0285714  - 0.1428571    0.3428571  - 0.2 !
! - 0.1428571    0.2142857  - 0.2142857    0.5 !
! - 0.2          0.5          0.1        - 0.1 !
!   0.3714286  - 0.3571429  - 0.0428571  - 0.1 !
-->//При умножении обратной матрицы на исходную,
-->//получилась матрица близкая к единичной.
-->inv(A)*A
 ans  =
      1.   - 1.110D-16    0.            0.
      0.     1.         - 5.551D-17     5.551D-17
      0.     0.           1.            1.388D-17
      0.     0.           6.939D-17     1.
-->//При попытке обратить вырожденную матрицу
-->//(определитель равен или близок к нулю),
-->//пользователь получит сообщение об ошибке.
-->B=[1 2 3;1 4 5;1 6 7];
```

26 Любой ненулевой вектор x, принадлежащий некоторому векторному пространству, для которого Ax=Lx, где L некоторое число, называется собственным вектором матрицы A, L соответствующим ему собственным значением матрицы A.

27 Обратной матрицей по отношению к данной матрице называется матрица того же типа, которая будучи умноженной как слева, так и справа на данную матрицу, в результате даст единичную матрицу. То есть при умножении `A` на `inv(A)` слева должна получиться единичная матрица.

28 Вычисление основано на методе LU разложения .

```
-->inv(B)
      !--error    19
      Problem is singular
```

Листинг 3.35

- `pinv(A[,tol])` вычисляет *псевдообратную матрицу*[29] для матрицы `A`, с точностью `tol` (необязательный параметр);

```
-->pinv(A)
 ans  =
   0.0285714      - 0.1428571      0.3428571       - 0.2
 - 0.1428571        0.2142857    - 0.2142857         0.5
 - 0.2              0.5            0.1             - 0.1
   0.3714286      - 0.3571429    - 0.0428571       - 0.1
```

Листинг 3.36

- `linsolve(A,b)` решает систему линейных алгебраических уравнений вида A*x+b=0.

```
-->//Решение системы линейных уравнений
-->//{x1+2x2-7=0; x1+x2-6=0}.
-->//Свободные коэффициенты вводятся как вектор-столбец
-->//и с учетом знаков.
-->A=[1 2;1 1];b=[-7;-6];
-->x=linsolve(A,b)
 x  =
      5.
      1.
-->//Результатом операции A*x+b является вектор достаточно
-->//близкий к нулю, это значит, что система решена верно.
-->A*x+b
 ans  =
   1.0D-14 *
      - 0.6217249
        0.0888178
-->//Решение системы {x1+x2-1=0; x1+x2-3=0}
-->A=[1 1;1 1]; b=[-1;-3];
-->//Система не имеет решений:
-->linsolve(A,b)
WARNING:Conflicting linear constraints!
 ans  =
     []
-->//Решение системы {3x1-x2-1=0; 6x1-2x2-2=0}.
-->//В случае, когда система имеет бесконечное
-->//множество решений,SCILAB выдаст одно из них.
-->A=[3 -1;6 -2];
-->b=[-1;-2];
-->x=linsolve(A,b)
 x  =
```

29 Если для матрицы pinv(A[,tol])=X выполняется условие A*X*A=A, X*A*X=X, а матрицы X*A и A*X эрмитовы, то вычисляемая матрица X *псевдообратная* к A. Вычисление базируется на методе сингулярного разложения и все сингулярные значения матрицы меньшие tol приравниваются нулю.

```
      0.3
    - 0.1
-->//Проверка не верна
-->A*x+b
 ans  =
   1.0D-15 *
      - 0.1110223
      - 0.2220446
```

Листинг 3.37

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

```
--> A=[3 -2 1 5;6 -4 2 7;9 -6 3 12]
A =
3-2     1     5
6-4     2     7
9-6     3     12
--> rref(A)
ans =
1.0000          -0.6667          0.3333     0
0               0                0                   1.0000
0               0                0                   0
```

Листинг 3.38

- `lu(M)` выполняет треугольное разложение матрицы M[30];

```
-->A=[2 -1 5;3 2 -5;1 1 -2]
 A  =
      2.   - 1.      5.
      3.     2.    - 5.
      1.     1.    - 2.
-->[L,U]=lu(A)
 U  =
      3.     2.           - 5.
      0.   - 2.3333333      8.3333333 !
      0.     0.             0.8571429 !
 L  =
      0.6666667      1.            0.
      1.             0.            0.
      0.3333333    - 0.1428571     1.
-->LU=L*U
 LU  =
      2.   - 1.      5.
      3.     2.    - 5.
1.    1.   - 2.
```

Листинг 3.39

30 M=€ ŁU, где L и U соответственно нижняя и верхняя треугольные матрицы , все четыре матрицы квадратные и одного порядка. Такие вычисления называют LU-разложеним.

- `qr(M)` выполняет разложение матрицы `M`[31] на ортогональную и верхнюю треугольную матрицу;

```
-->[Q,R]=qr(A)
 R  =
- 3.7416574  - 1.3363062    1.8708287
  0.         - 2.0528726    7.0632734
  0.           0.           0.7811335
 Q  =
- 0.5345225    0.8350668    0.1301889
- 0.8017837  - 0.4523279  - 0.3905667
-  0.2672612  - 0.3131501    0.9113224
-
-->Q*R
 ans  =
2.  - 1.    5.
3.    2.  - 5.
1.    1.  - 2.
```

Листинг 3.40

- `svd(M)` выполняет *сингулярное разложение*[32] матрицы `M` размером `n×m`; результатом работы функции может быть либо сингулярное разложение, либо вектор, содержащий сингулярные значения матрицы;

```
-->[U,S,V]=svd(A)
 V  =
- 0.1725618    0.9641403  - 0.2016333
- 0.3059444    0.1421160    0.9413825
  0.9362801    0.2241352    0.2704496
 S  =
7.8003308    0.           0.
0.           3.6207331    0.
0.           0.           0.2124427
 U  =
  0.5951314    0.8028320    0.0357682
- 0.7449652    0.5678344  - 0.3501300
- 0.3014060    0.1817273    0.9360180
-->U*S*V'
 ans  =
2.  - 1.    5.
3.    2.  - 5.
1.    1.  - 2.
-->s=svd(A)
 s  =
7.8003308
```

31 M=QCR, где Q ортогональная матрица , а R верхняя треугольная матрицы .Этот процесс называют QR-разложеним.

32 M=U⋅S⋅VT, где U и V ортогональные матрицы размером m×m и n×n соответственно , а S диагональная матрица, на диагонали которой расположены сингулярные числа матрицы M

```
3.6207331
0.2124427
```

Листинг 3.41

- `kernel(M[,tol[,fl]])` определение *ядра матрицы*[33] `M`, параметры `tol` и `fl` являются не обязательными. Первый задает точность вычислений, второй используемый при вычислении алгоритм и принимает значения `'qr'` или `'svd'`.

```
-->A=[4 1 -3 -1;2 3 1 -5;1 -2 -2 3]
 A  =
4.    1.  - 3.  - 1.
2.    3.    1.  - 5.
1.  - 2.  - 2.    3.
-->X=kernel(A)
 X  =
0.3464102
0.5773503
0.4618802
0.5773503
```

Листинг 3.42

3.4 Символьные матрицы и операции над ними

В SCILAB можно задавать символьные матрицы, то есть матрицы, элементы которых имеют строковый тип. При этом необходимо помнить, что строковые элементы должны быть заключены в двойные или одинарные кавычки.

```
-->M=['a' 'b';'c' 'd']
 M  =
a  b
c  d
-->P=['1' '2';'3' '4']
 P  =
1  2
3  4
```

Листинг 3.43

Символьные матрицы можно складывать (результат сложения конкатенация соответствующих строк) и транспонировать:

```
-->M+P
 ans  =
a1  b2
c3  d4
-->M'
```

33 Ядро матрицы это множество векторов X. Поиск ядра матрицы сводится к решению однородной системы линейных уравнений AX=0. Если при вызове функции X=kernel(A) матрица X окажется не пустой, то действительно AX=0.

```
 ans  =
a  c
b  d
```

Листинг 3.44

Кроме того операции сложения и умножения можно проводить над отдельными элементами символьных матриц при помощи функций `addf(a,b)` и `mulf(a,b)`:

```
-->addf(M(1,1),P(2,2))
 ans  =
 a+4
-->mulf(M(1,2),P(2,1))
 ans  =
 b*3
```

Листинг 3.45

3.5 Решение систем линейных алгебраических уравнений

Система *m* уравнений с *n* неизвестными вида:

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

называется *системой линейных алгебраических уравнений* (СЛАУ), причем x_j неизвестные , a_{ij} коэффициенты при неизвестных , b_i свободные коэффициенты (i=1..*m*, *j*=1..*n*). Система из *m* линейных уравнений с *n* неизвестными может быть описана при помощи матриц: $A{\cdot}x=b$, где *x* вектор неизвестных , *A* матрица коэффициентов при неизвестных или матрица системы, *b* вектор свободных членов системы или вектор правых частей . Совокупность всех решений системы (x_1, x_2,..., x_n), называется множеством решений или просто *решением системы.*

ЗАДАЧА 3.1.

Решить СЛАУ при помощи правила Крамера:

$2x_1+ x_2 \quad 5x_3+ x_4= \ 8,$

$x_1 \quad 3x_2 \qquad 6x_4= \ 9,$

$2x_2 \quad x_3+2x_4=-5,$

$x_1+4x_2 \quad 7x_3+6x_4= 0.$

Правило Крамера заключается в следующем. Если определитель $\Delta=det$A матрицы системы из *n* уравнений с *n* неизвестными $A{\cdot}x=b$ отличен от нуля, то система имеет единственное решение x_1, x_2,..., x_n, определяемое по формулам Крамера: $x_i=\Delta_i/\Delta$, где Δ_i определитель матрицы, полученной из матрицы системы *A* заменой *i* го столбца столбцом свободных членов *b*. Текст файла сценария с решением задачи по формулам Крамера :

```
//Матрица коэффициентов:
A=[2 1 -5 1;1 -3 0 -6;0 2 -1 2;1 4 -7 6];
b=[8;9;-5;0];   //Вектор свободных коэффициентов
A1=A;A1(:,1)=b; //Первая вспомогательная матрица
A2=A;A2(:,2)=b; //Вторая вспомогательная матрица
A3=A;A3(:,3)=b; //Третья вспомогательная матрица
A4=A;A4(:,4)=b; //Четвертая вспомогательная матрица
D=det(A);       //Главный определитель
//Определители вспомогательных матриц:
```

```
d(1)=det(A1); d(2)=det(A2); d(3)=det(A3); d(4)=det(A4);
x=d/D          //Вектор неизвестных
P=A*x-b        //Проверка
```

Листинг 3.46

Результаты работы файла-сценария:

```
-->exec('C:\scilab-4.1.1\kramer.sce');disp('exec done');
 x  =
    3.
  - 4.
  - 1.
    1.
 P  = 1.0D-14 *
         0.1776357
         0.
       - 0.0888178
         0.1554312
 exec done
```

Листинг 3.47

ЗАДАЧА 3.2.

Решить СЛАУ из задачи 3.1 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 $x=A^{-1}\cdot b$.

Текст файла сценария и результаты его работы :

```
//Матрица и вектор свободных коэффициентов системы:
A=[2 1 -5 1;1 -3 0 -6;0 2 -1 2;1 4 -7 6];b=[8;9;-5;0];
x=inv(A)*b      //Решение системы
//Результаты работы файла-сценария:
--> x  =
           3.
           - 4.
           - 1.
           1.
```

Листинг 3.48

ЗАДАЧА 3.3.

Решить систему линейных уравнений методом Гаусса:

$2x_1 \quad x_2+5x_3=0,$

$3x_1+2x_2 \quad 5x_3=1,$

$x_1+ x_2 \quad 2x_3=4.$

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 матрица6 системы путем элементарных преобразований (перестановка уравнений системы, умножение уравнений на число отличное от нуля и сложение уравнений) приводится к ступенчатому виду. На втором этапе (обратный ход) ступенчатую матрицу преобразовывают так, чтобы в первых n столбцах получилась единичная матрица. Последний, *n*+1 столбец этой матрицы содержит решение системы линейных уравнений. Далее приведен текст файла-сценария и результаты его работы:

```
//Матрица и вектор свободных коэффициентов системы:
A=[2 -1 1;3 2 -5;1 3 -2]; b=[0;1;4];
//Приведение расширенной матрицы к треугольному виду:
C=rref([A b]);
//Определение размерности расширенной матрицы:
[n,m]=size(C);  //m- номер последнего столбца матрицы C
//Выделение последнего столбца из матрицы C:
x=C(:,m)        //x - решение системы
//Результаты работы программы:
--> x  =
    0.4642857
    1.6785714
    0.75
```

Листинг 3.49