Тема 3. Операции с векторами и матрицами

Под ***вектором*** в MatLAB понимается одномерный массив чисел, а под ***матрицей*** – двумерный массив.

При этом по умолчанию предполагается, что любая заданная переменная является вектором или матрицей. Например, отдельное заданное число система воспринимает как матрицу размером (1*1), а вектор-строку из N элементов - как матрицу размером (1*N).

MATLAB допускает использование многомерных массивов (Эвклидово пространство).

3.1. Ввод векторов и матриц

Ввод **векторов** осуществляется в виде:

V=[x1 x2 x3]

где V – имя вектора,

x1 x2 x3 – значения элементов вектора, заключенные в *квадратные* скобки и отделенные друг от друга *пробелами* или *запятыми*.

Например, запись строки V = [1.2 -0.3 1.2e-5]

Длинный вектор можно вводить частями, которые потом объединять с помощью операции *объединения векторов в строку :*

V = [v1 v2]

Например, v1 = [1 2 3]; v2 = [4 5 6]; V = [v1 v2]

Так вводятся векторы-строки. *Вектор-столбец* вводится аналогично, но значения элементов отделяются знаком ";"

Введение арифметической прогрессии

Для формирования упорядоченных числовых последовательностей используется оператор **":"**

Если обозначить: nz – начальное значение этой прогрессии (значение первого элемента вектора); kz – конечное значение прогрессии (значение последнего элемента вектора); h – разность прогрессии (шаг), то вектор можно ввести с помощью короткой записи

$$V = nz : h : kz$$

Например, V = - 0.1 : 0.3 :1.4

Если шаг прогрессии не указан, то он по умолчанию принимается равным *единице*.

Например, команда

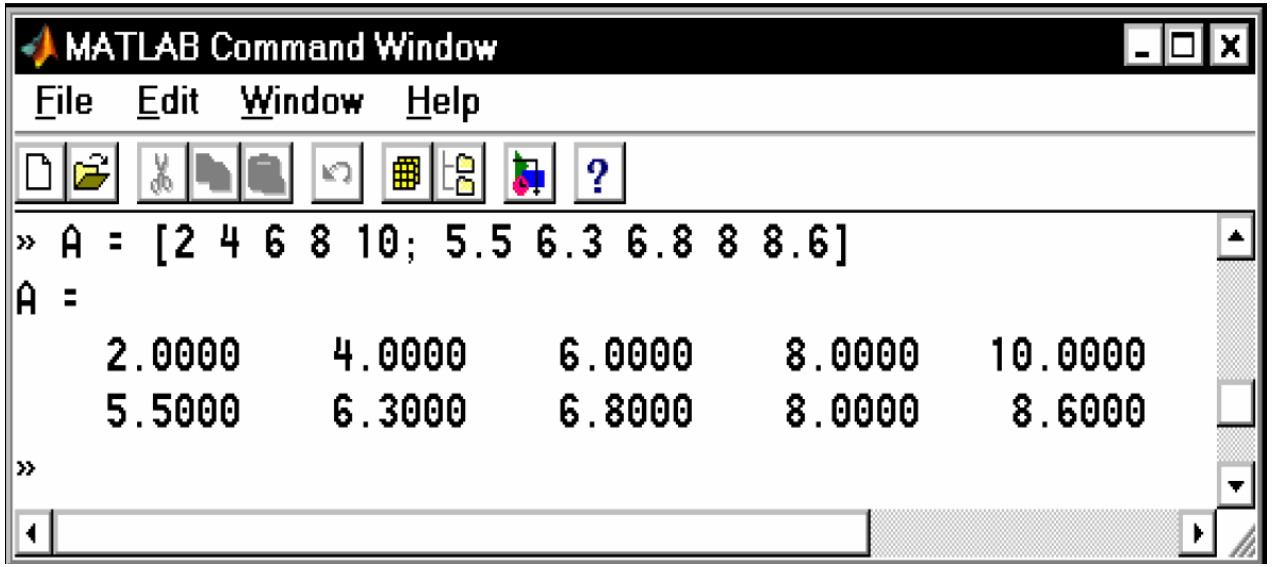
```
>>-2.1:5
```

приводит к формированию такого вектора

```
ans = -2.1000 -1.1000 -0.1000 0.9000
  1.9000 2.9000 3.9000 4.9000
```

Ввод значений элементов **матрицы** осуществляется в квадратных скобках, *по строкам*. При этом элементы строки матрицы один от другого отделяются пробелом или запятой, а строки одна от другой отделяются знаком ";"

Пример:



Возможен также ввод элементов матриц и векторов в виде арифметических выражений, содержащих любые доступные системные функции

Пример: V=[2+2/(3+4) exp(5) sqrt(10)]

3.2. Формирование векторов и матриц

Некоторые функций MATLAB, позволяющие формировать вектора и матрицы определенного вида:

- ***zeros(M,N)*** – создает матрицу размером (M*N) с нулевыми элементами;
- ***ones(M,N)*** – создает матрицу размером (M*N) с единичными элементами;
- ***eye(M,N)*** – создает единичную матрицу размером (M*N), т. е. с единицами по главной диагонали и остальными нулевыми элементами;
- ***rand(M,N)*** – создает матрицу размером (M*N) из случайных чисел, равномерно распределенных в диапазоне от 0 до 1;
- ***randn(M,N)*** – создает матрицу размером (M*N) из случайных чисел, распределенных по нормальному (гауссовому) закону с нулевым математическим ожиданием и стандартным (среднеквадратичным) отклонением, равным единице;
- ***magic(N)*** – создает матрицу размером (N*N) у которой сумма всех строк, столбцов и диагоналей равна одному и тому же числу;

В MatLAB предусмотрено несколько функций, которые позволяют формировать матрицу на основе другой (заданной) или используя некоторый заданный вектор. К таким функциям принадлежат:

- ***fliplr(A)*** – формирует матрицу, переставляя столбцы известной матрицы A относительно вертикальной оси, например:

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

- ***flipud(A)*** – переставляет строки заданной матрицы A относительно горизонтальной оси;
- ***rot90(A)*** – формирует матрицу путем "поворота" заданной матрицы A на 90 градусов против часовой стрелки;

- ***reshape(A,m,n)*** – образует матрицу размером (m*n) путем выборки элементов заданной матрицы A по столбцам и последующему распределению этих элементов по 'n' столбцам, каждый из которых содержит 'm' элементов; при этом число элементов матрицы A должно равняться m*n, например:

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

- ***tril(A)*** – образует нижнюю треугольную матрицу на основе матрицы A путем обнуления ее элементов выше главной диагонали;
- ***triu(A)*** – образует верхнюю треугольную матрицу на основе матрицы A путем обнуления ее элементов ниже главной диагонали;

- Процедура ***diag(x)*** - формирует или извлекает диагональ матрицы.

Если ***x*** - вектор, то функция ***diag(x)*** создает квадратную матрицу с вектором ***x*** на главной диагонали:

» diag(V)

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

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

» diag(V, -1)

Если ***x*** - матрица, то функция ***diag*** создает вектор-столбец, который состоит из элементов главной диагонали заданной матрицы ***x***, например, для матрицы A:

» diag(A)

```
ans =
 1
 8
15
```

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

3.3. Извлечение и вставка частей матриц

Обращение к любому элементу заданной матрицы осуществляется путем указания (*в скобках, через запятую*) после имени матрицы двух целых положительных чисел, которые определяют соответственно *номера строки и столбца* матрицы, на пересечении которых расположен этот элемент.

Пусть имеем некоторую матрицу A:

>> A = [1 2 3 4; 5 6 7 8; 9 10 11 12]

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

Тогда получить значение элемента этой матрицы, расположенного на пересечении второй строки с третьим столбцом, можно следующим образом:

>> A(2,3)

```
ans = 7
```

Если нужно, наоборот, установить на это место некоторое число, например, π , то это можно сделать так:

>> A(2, 3) = pi;

Если нужно извлечь из матрицы строку или столбец, либо создать меньшую матрицу из большей, или, наоборот, вставить меньшую матрицу таким образом, чтобы она стала определенной частью матрицы большего размера, то тогда необходимо применять оператор двоеточие (" : ").

Пример: пусть нужно создать вектор V1, состоящий из элементов третьего столбца матрицы A. Для этого произведем такие действия:

>> V1 = A(:, 3)

```
V1 =
3.0000
3.1416
11.0000
```

Чтобы создать вектор V2, состоящий из элементов второй строки матрицы A, необходимо:

>> V2 = A(2, :)

```
V2 = 5.0000 6.0000 3.1416 8.0000
```

Допустим, что необходимо из матрицы A образовать матрицу B размером (2*2), которая состоит из элементов левого нижнего угла матрицы A. Тогда делают так:

>> B = A(2:3, 1:2)

```
B =
5   6
9  10
```

Аналогично можно вставить матрицу B в верхнюю середину матрицы A:

>> A(1:2,2:3)=B

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

Как видно, для этого *вместо указания номеров элементов матрицы можно указывать диапазон изменения этих номеров путем указания нижней и верхней границ, разделяя их двоеточием.*

Эти операции очень удобны для формирования матриц, большинство элементов которых одинаковы, в частности, так называемых разреженных матриц, которые состоят, в основном, из нулей, за исключением отдельных элементов.

Для примера рассмотрим формирование разреженной матрицы размером (5*7) с единичными элементами в ее центре:

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

```
A =
     0     0     0     0     0     0     0
     0     0     1     1     1     0     0
     0     0     1     1     1     0     0
     0     0     1     1     1     0     0
     0     0     0     0     0     0     0
```

"Растянуть" матрицу (A) в единый вектор (V) можно с помощью обычной записи "V = A(:)". При этом создается вектор-столбец с количеством элементов (m*n), в котором столбцы заданной матрицы размещены сверху вниз в порядке возрастания:

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

Удаление столбцов и строк матриц

Для удаления отдельных столбцов и строк матрицы используют пустые квадратные скобки [ ].

Пример:

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

Удалим второй столбец используя оператор “:”.

>>M(:,2)=[]

Аналогично можно удалить строку

>>M(2,:)=[]

Объединение малых матриц в большую

Операция объединения малых матриц в большую называется ***конкатенация***.

Существует *горизонтальная* и *вертикальная* конкатенация.

Горизонтальная конкатенация – это объединение нескольких матриц-блоков A1, A2,... AN с одинаковым количеством *строк*.

$$A = [A1, A2, \ldots, AN]$$

Пример:

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

Вертикальная конкатенация – это объединение нескольких матриц-блоков при условии, что все составные блоки-матрицы имеют одинаковое количество *столбцов*. При этом, для отделения блоков вместо запятой используется ";".

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

Пример:

```
>> B1 = [1 2 3 4 5];
>> B2 = [6 7 8 9 10; 11 12 13 14 15];
>> B3 = [17 18 19 20 21];
>> B = [B1; B2; B3]
B =
 1  2  3  4  5
 6  7  8  9 10
11 12 13 14 15
17 18 19 20 21
```

3.4. Действия над векторами

Различают две группы действий над векторами:

а) векторные действия – т. е. такие, которые предусмотрены векторным исчислением в математике;

б) *действия по преобразованию элементов – это действия, которые преобра*зуют элементы вектора, но не являются операциями, разрешенными математикой.

Векторные действия над векторами

Сложение векторов. Как известно, суммироваться могут только векторы одинакового типа (т. е. такие, которые являются или векторами-строками, или векторами-столбцами), имеющие одинаковую длину (т. е. одинаковое количество элементов). Если X и Y - именно такие векторы, то их сумму Z можно получить, введя команду Z = X + Y, например:

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

Аналогично осуществляется ***вычитание*** векторов.

- ***Транспонирование вектора*** осуществляется применением знака апострофа, который записывается сразу за записью имени вектора, который нужно транспонировать. Например:

» x’

- ***Умножение вектора на число*** осуществляется с помощью знака арифметического умножения ' * ' таким образом: Z = X*r или Z = r*X, где r – некоторое действительное число.
- ***Умножение двух векторов*** определено в математике только для векторов одинакового размера (длины) и лишь тогда, когда один из векторов-множителей строка, а второй - столбец. Иначе говоря, если векторы X и Y являются строками, то математическое смысл имеют лишь две формы умножения этих векторов: U = X' * Y и V = X * Y'. При этом в первом случае результатом будет квадратная матрица, а во втором - число.

Пример

» x = [1 2 3] ; y = [4 5 6];

» v = x' * y

```
v =
 4   5   6
 8  10  12
12  15  18
```

» v = x * y'

```
v = 32
```

Поэлементное преобразование векторов

В языке MatLAB предусмотрен ряд операций, которые преобразуют заданный вектор в другой того же размера и типа, но не являются операциями с вектором как математическим объектом. К таким операциям относятся, например, *все элементарные математические функции* зависящие от одного аргумента.

Кроме этих операций в MatLAB предусмотрено несколько операций поэлементного преобразования, осуществляемых с помощью знаков обычных арифметических действий. *Эти операции применяются к векторам одинакового типа и размера.* Результатом их есть вектор того же типа и размера.

Добавление (отнимание) числа к (из) каждому элемента вектора. Осуществляется с помощью знака ' + ' (' - ').

Поэлементное умножение векторов. Проводится с помощью совокупности знаков ' .* ', которая записывается между именами перемножаемых векторов. В результате получается вектор, каждый элемент которого является произведением соответствующих элементов векторов - "сомножителей".

Поэлементное деление векторов. Осуществляется с помощью совокупности знаков './ ' или '.\ '.

Поэлементное возведение в степень. Осуществляется с помощью совокупности знаков '.^ ‘.

Пример 1:

» x = [1,2,3,4,5]; y = [-2,1,4,0,5];

» disp(x. /y)

Пример 2: пусть нужно вычислить значения функции:

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

при значениях аргумента ***x*** от 0 до 10 с шагом 1.

Вычисление массива значений этой функции в указанных условиях можно осуществить с помощью лишь двух простых операторов :

» a = 3; h = 0.5; x = 0:10;

» y = a * exp(-h*x) . * sin(x)

```
y =
Columns 1 through 7
0 1.5311 1.0035 0.0945 -0.3073 -0.2361 -0.0417
Columns 8 through 11
0. 0595 0. 0544 0. 0137 -0. 0110
```

Аналогично осуществляется ***поэлементное преобразование матриц*** , при этом матрицы должны быть одинакового размера.

3.5. Матричные действия над матрицами

К матричным действиям над матрицами относят такие операции, которые используются в матричном исчислении в математике и не противоречат ему.

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

- *при сложении или вычитании матрицы должны иметь одинаковые размеры;*
- *при умножении матриц количество столбцов первой матрицы должно совпадать с количеством строк второй матрицы.*

- Функция ***обращения матрицы - inv(A)*** – вычисляет матрицу, обратную заданной матрице A. Исходная матрица A должна быть квадратной, а ее определитель не должен равняться нулю.
- ***Возведение матрицы в целую степень***: A^*n*. При этом матрица должна быть квадратной, а **n** - целым (положительным или отрицательным) числом.
- Оригинальными в языке MatLAB являются две новые, неопределяемые в математике функции ***деления матриц.*** При этом вводятся понятие ***деления матриц слева направо*** и ***деление матриц справа налево.***
- Операция B / A эквивалентна последовательности действий B * ***inv(A).***

Ее удобно использовать для решения матричного уравнения:

$$X * A = B.$$

Аналогично операция A\B равносильна совокупности операций ***inv(A)*B,*** которая представляет собой решение матричного уравнения:

$$A * X = B.$$

Пример: необходимо найти корни системы линейных алгебраических уравнений:

$$x_1 + 2x_2 + 3x_3 = 14$$
$$2x_1 - x_2 - 5x_3 = -15$$
$$x_1 - x_2 - x_3 = -4$$

В среде MatLAB это можно сделать таким образом:

```
» A = [ 1 2 3; 2 -1 -5; 1 -1 -1]
A =
1  2  3
2 -1 -5
1 -1 -1
» B = [ 14;-15;-4]
B =
14
-15
-4
» x = A \ B
x =
1
2
3
```