**Лабораторная работа №2**

**Матрицы**

***План занятия***

1. Ввод матриц. Различные способы задания матриц.
2. Объединение матриц. Удаление строк и столбцов.
3. Элементарные операции над матрицами.
4. Матричные операции.
5. Применение функций обработки данных к матрицам.
6. Поэлементные операции над матицами.

# *Ввод матриц*

 При вводе матриц необходимо соблюдать основные условия:

* отделять элементы строки пробелами или запятыми;
* использовать точку с запятой для обозначения окончания каждой строки, заключать весь список элементов квадратными скобками *[ ]*;
* при обращении к элементам массива использовать круглые скобки *( )*.

 Матрицы в *Matlab* можно вводить несколькими способами:

* вводить полный список элементов;
* загружать матрицы из внешних файлов;
* генерировать матрицы, используя встроенные функции;
* создавать матрицы с помощью ваших собственных функций в *М* – файлах.

 Явное определение матриц

*>>A=[3 1 1; 4 5 6]*

*A=*

 *3 1 1*

 *4 5 6*

 Посмотреть переменные рабочей среды можно с помощью команд *who* или *whos*. При этом будет выведена информация

*A 2×3 48 double array*

 То есть *A* представляет собой двумерную матрицу размером *2×3*, занимающую 48 байтов.

 При вводе чисел в экспоненциальной форме (например, *2.34e-9*), не следует использовать пробелы. Ввод больших матриц лучше выпол­нять с помощью *М-*файлов, в которых легко находить и исправлять ошибки. Создание *М-*файлов будет рассмотрено на занятии №6.

 Ссылки на отдельные элементы матриц и векторов осуществляются с помощью индексов в круглых скобках обычным образом. Элемент в строке *i* и столбце *j* матрицы *A* обозначается *A(i,j)*. Например, *А(2, 3)=6.*

 На элементы матрицы *A* можно ссылаться, используя единствен­ный индекс: *A(k)*. В этом случае эта матрица рассматривается как один длинный вектор-столбец, сформированный из столбцов исходной матрицы. В приведенной матрице элемент *6* может быть вызван как *A(2,3),* либо *A(6)*. И то и другое правильно.

# *Генерирование матриц*

*Matlab* имеет специальные функции, которые создают основные матрицы:

*zeros* – все элементы матрицы - нули

*eye* – единичная матрица

*ones* – все элементы матрицы - единицы

*rand* – равномерное распределение случайных элементов

*randn* – нормальное распределение случайных элементов.

 Например, команда *rand(n)* создает квадратную матрицу *nxn*, каждый элемент которой – случайное число с равномерным распределением в диапазоне *[0 1]*, в то время как команда *rand(m, n)* создает такую же матрицу размера *mxn*. Матрицы могут быть сгенерированы также с помощью цикла *for* (см. занятие №8).

 Некоторые примеры:

*>> Z = zeros(2,3)*

*Z =*

 *0 0 0*

 *0 0 0*

*>> Q = eye(3,4)*

*Q =*

 *1 0 0 0*

 *0 1 0 0*

 *0 0 1 0*

*>> W = 5\*ones(3,3)*

*W =*

 *5 5 5*

 *5 5 5*

 *5 5 5*

*>> N = fix(10\*rand(1,9))*

*N =*

 *9 2 6 4 8 7 4 0 8*

*>> R = randn(4,4)*

*R =*

 *-0.4326 -1.1465 0.3273 -0.5883*

 *-1.6656 1.1909 0.1746 2.1832*

 *0.1253 1.1892 -0.1867 -0.1364*

 *0.2877 -0.0376 0.7258 0.1139*

 Часто возникает необходимость создания диагональных матриц, т. е. мат­риц, у которых все внедиагональные элементы равны нулю. Функция *diag* формирует диагональную матрицу из вектора или вектор-строки, располагая их элементы по диагонали матрицы:

*>> d = [1; 2; 3; 4];*

*>> D = diag(d)*

*D =*

 *1 0 0 0*

 *0 2 0 0*

 *0 0 3 0*

 *0 0 0 4*

 Для заполнения не главной, а побочной диагонали предусмотрена возмож­ность вызова функции *diag* с двумя аргументами. В этом случае второй ар­гумент означает, насколько побочная диагональ отстоит от главной, а его знак указывает на направление, плюс – вверх, минус – вниз от главной диагонали:

*>> d = [1; 2];*

*>> D = diag(d, 2)*

*D =*

 *0 0 1 0*

 *0 0 0 2*

 *0 0 0 0*

 *0 0 0 0*

*>> D = diag(d, -2)*

*D =*

 *0 0 0 0*

 *0 0 0 0*

 *1 0 0 0*

 *0 2 0 0*

 Функция *diag* служит и для выделения диагонали матрицы в век­тор, например

*>> А = [10 1 2; 1 20 3; 2 3 30]*

*А =*

 *10 1 2*

 *1 20 3*

 *2 3 30
>> d = diag(A)*

*d =*

 *10*

 *20*

 *30*

 *Matlab* обладает встроенной функцией, которая создает маги­ческий квадрат почти любого размера (в магическом квадрате сумма элемен­тов строк равна сумме элементов столбцов и равна сумме элементов главной и побочной диагоналей). Не удивительно, что эта функция назы­вается *magic*.

*>> B= magic(4)*

*В =*

 *16 2 3 13*

 *5 11 10 8*

 *9 7 6 12*

 *4 14 15 1*

# *Загрузка матриц*

 Команда *load* считывает двоичные файлы, содержащие матрицы, созданные в *Matlab* ранее, или текстовые файлы, содержащие числен­ные данные. Тексто­вые файлы должны быть сформированы в виде прямоугольной таблицы чисел, отделенных пробелами, с равным количеством элементов в каждой строке. Пусть вне *Matlab* создан текстовой файл, содержащий 4 строки:

*16 2 3 13*

*5 11 10 8*

*9 7 6 12*

*4 14 15 1*

который сохранен под именем *magik.txt* в текущем каталоге *Matlab* *work*. Тогда команда

*>>load magik.txt*

прочитает этот файл и создаст переменную *magik*, содержащую дан­ную матрицу. Значение данной матрицы можно присвоить некоторой переменной:

*>>D=load(‘magik.txt’);*

*>> d=D;*

 Для сохранения данных рабочей области в текстовом файле применя­ется команда *save*:

*>> save ‘magcopy’ d –ascii*

 По этой команде значения переменной *d* будет сохранено в файле *‘magcopy’*. Параметр *–ascii* означает запись в текстовом формате. Для записи результата в файл с двойной точностью следует использовать команду

*>> save ‘magcopy’ d –ascii –double*

# *Объединение*

 Объединение - это процесс соединения маленьких матриц для создания боль­ших. Пара квадратных скобок - это оператор объедине­ния. Пользуясь матрицей *D* (магического квадрата *4×4*), можно сформировать новую матрицу:

*>> В = [D; D+32; D+48; D+16]*

 Результатом будет матрица *8×8*, получаемая соединением четырех подматриц

*16 2 3 13 48 34 35 45*

*5 11 10 8 37 43 42 40*

*9 7 б 12 41 39 38 44*

*4 14 15 1 36 46 47 33*

*64 50 51 61 32 18 19 29*

*53 59 58 56 21 27 26 24*

*57 55 54 60 25 23 22 28*

*52 62 63 49 20 30 31 17*

# *Удаление строк и столбцов*

 Удалять строки и столбцы матрицы можно, используя пару пустых квадрат­ных скобок. Пусть матрица определена следующим образом:

*>> X = D;*

и требуется удалить второй столбец матрицы *X*. Для этого нужно произвести следующее действие

*>> X(:,2)=[ ]*

 Эта операция изменит *X* следующим образом:

*X =*

 *16 3 13*

 *5 10 8*

 *9 6 12*

 *4 15 1*

# *Сложение, вычитание, умножение, транспонирование и возведение в степень*

 При использовании матричных операций следует помнить, что для сложе­ния или вычитания матрицы должны быть одного размера, а при перемно­жении число столбцов первой матрицы обязано равняться числу строк вто­рой матрицы. Сложение и вычитание матриц, так же как чисел и векторов, осуществляется при помощи знаков плюс и минус.

 Пусть

*>>A=[1 2 3;4 5 6];*

*>> B=[5 6 2;8 9 0];*

*>>S=A+B*

*S =*

 *6 8 5*

 *12 14 6*

*>> R=A-B*

*R =*

 *-4 -4 1*

 *-4 -4 6*

 Необходимо следить за совпадением размерности, иначе будет выдано сообщение об ошибке:

*>> S = А+В*

*??? Error using ==> + MATrix dimensions must agree.*

 Для умножения матриц предназначена звездочка:

*>> C=[1 2 ; 3 4;1 2];*

*>> Р =A\*С*

*P =*

 *10 16*

 *25 40*

 Умножение матрицы на число тоже осуществляется при помощи звездочки, причем умножать на число можно как справа, так и слева:

*>> Р = А\*3*

*Р =*

 *3 6 9*

 *12 15 18*

*>> Р = 3\*А*

*P =*

 *3 6 9*

 *12 15 18*

 Транспонирование матрицы, так же как и вектора, производится при помощи *.'*, а символ *'* означает комплексное сопряжение. Для вещественных матриц эти операции приводят к одинаковым результа­там:
*>> B'*

*ans =*

*5 8*

*6 9*

*2 0*

*>> B.'*

*ans =*

*5 8*

*6 9*

*2 0*

 Двоеточие “:” - это один из наиболее важных операторов *Matlab*. Так *А(1:k, j)* - это первые *k* элементов *j*-гo столбца матрицы *А*. Функция *sum(A(1:2, 3))* вычисляет сумму элементов первых двух строк третьего столбца. Двоето­чие позволяет обращаться ко всем элементам в строке и столбце матрицы, а слово *end* - к последней строке или столбцу. Так *sum(A(:,end))* вычисляет сумму элементов в последнем столбце мат­рицы *А*.

 Сопряжение и транспонирование матриц, содержащих комплекс­ные числа, приводят к созданию разных матриц:

*>> К = [1-i, 2+3i; 3-5i, 1-9i]*

*К =*

 *1.0000 - 1.0000i 2.0000 + 3.0000i*

 *3.0000 - 5.0000i 1.0000 - 9.0000i*

*>> К'*

 *ans =*

 *1.0000 + 1.0000i 3.0000 + 5.0000i*

 *2.0000 - 3.0000i 1.0000 + 9.0000i*

*>> К.'*

*ans =*

 *1.0000 - 1.0000i 3.0000 - 5.0000i*

 *2.0000 + 3.0000i 1.0000 - 9.0000i*

 Возведение квадратной матрицы *D* в целую степень производится с использо­ванием оператора *^*:

*>> D2 = D^2*

 Проверьте полученный результат, умножив матрицу саму на себя.

 При выполнении операций над матрицами необходимо учитывать приоритет операций: сначала выполнять транспонирование, потом возведение в степень, затем умножение, а сложение и вычитание производить в последнюю очередь.

# *Перемножение матрицы и вектора*

 Например, вычисление выражения



можно осуществить следующим образом:

*>> а = [1 3 -2];*

*>> В = [2 0 1; -4 8 -1; 0 9 2];*

*>> с = [-8;3;4];*

*>> а\*В\*с*

*ans = 74*

# *Решение систем линейных уравнений*

 Ставится задача решения системы, состоящей из трех уравнений с тремя неиз­вестными:

 *1.2x1+0.3x2 - 0.2х3 =1.3;*

 *0.5х1 +2.1х2 +1.3x3 =3.9;*

*-0.9x1 +0.7х2 +5.6х3 =5.4*

 Введите матрицу системы в массив *A*, для вектора правой части используйте массив *b*. Решите систему при помощи символа \

*>> х = А\b*

*1. 0 000*

*1. 0000*

*1. 0000*

 Проверьте, правильный ли получился ответ, умножив *А* на *х*.

# *Применение функций обработки данных к матрицам*

 По функции *sum*, примененной к матрице, *Matlab* позволяет вычислить вектор-строку, длина ко­торой равна числу столбцов матрицы, а каждый элемент является суммой элементов соответствующего столбца матрицы, напри­мер:

*>> М = [1 -2 -4*

 *3 -6 4*

 *2 -2 0];*

*>> s = sum(M)*

 *6 -10 0*

 Функция *sum* по умолчанию вычисляет сумму по столбцам, изме­няя первый индекс массива при фиксированном втором. Для того чтобы производить суммирование по строкам, необходимо вызвать функцию *sum* с двумя аргу­ментами, указав индекс, по которому следует суммировать:

*>> s2 = sum(M, 2)*

 *s2 =*

 *-5*

 *1*

 *0*

 Аналогично работает и функ­ция *prod*:

*>> р = prod(M)*

*p =*

 *6 -24 0*

*>> p2 = prod(M, 2)*

*р2 =*

 *8*

 *72*

 *0*

 Функция *sort* упорядочивает элементы каждого из столбцов мат­рицы в по­рядке возрастания. Вызов *sort* со вторым аргументом, рав­ным двум, приво­дит к упорядочению элементов строк:

*>> МС = sort(M)*

*МС =*

 *1 -6 -4*

 *2 -2 0*

 *3 -2 4*

*>> MR = sort(M, 2)*

*MR =*

 *-4 -2 1*

 *-6 3 4*

 *-2 0 2*

 Функции *max* и *min* вычисляют вектор-строки, содержащие соответственно макси­мальные или минимальные элементы в столбцах матрицы:

*>> mx = max(М)*

*mх =*

 *3 -2 4*

*>> mn = min (M)*

*mn =*

 *1 -6 -4*

 Для того чтобы узнать не только значения максимальных или минимальных элементов, но и их номера в столбцах, следует вызвать *max* или *min* двумя выходными параметрами:

*>> [mx, k] = max (М)*

*mх =*

 *3 -2 4*

*k =*

 *2 1 2*

*>> [mn, n] = min(M)*

*mn =*

 *1 -6 -4*

*n =*

 *1 2 1*

 Если аргументами функций *min* или *max* являются числа, то возвращает­ся минимальное или максимальное из этих чисел.

 Для нахождения максимума или минимума не по столбцам мат­рицы, а по строкам предусмотрена следующая форма вызова со вто­рым аргументом - пустым массивом:

*>> maх = mах(M, [ ], 2)*

*maх =*

 *1*

 *4*

 *2*

 Более подробно про обработку матричных данных можно узнать, выведя список всех встроенных функции обработки данных командой *help* *datafun*, а затем посмотреть информацию о нужной функ­ции, например *help max.*

 Определитель матрицы находится по вызову функции *det*, ранг – *rank*.

# *Поэлементные операции с матрицами*

 Даны две матрицы:

 

 Умножение каждого элемента одной матрицы на соответствующий элемент другой производится при помощи оператора *.\**

*>> С = А.\*В*

*C=*

*-2 10 -8*

*21 -12 -45*

 Для деления элементов первой матрицы на соответствующие элементы вто­рой используется *./*, а для деления элементов второй матрицы на соответст­вующие элементы первой служит *.\*

*>> R1 = А./В*

*R1 =*

*-2.0000 2.5000 -0.1250*

*0.4286 -1.3333 -1.8000*

*>> R2 = А.\В*

*R2 =*

*-0.5000 0.4000 -8.0000*

*2.3333 -0.7500 -0.5556*

 Поэлементное возведение в степень осуществляется при помощи оператора .^

*>> Р = А.^2*

*Р =*

 *4 25 1*

 *9 16 81*

# Задание

1. Ввести матрицу *А*.
2. Найти *A'*.
3. Выполнить инвертирование *inv(A),* проверить тождество.
4. Создать диагональную матрицу, выделить ее диагональ.
5. Заполнить и записать в файлы матрицы:



1. Создать магический квадрат. Проверить свойство: сумма элементов любой строки совпадает с суммой элементов любого столбца и диаго­нали.