

ЛАБ.РАБОТА №2. MATLAB - ЭТО МАТРИЦЫ.

СОЗДАНИЕ МАТРИЦ И МАНИПУЛЯЦИЯ ИХ ЭЛЕМЕНТАМИ

На первых шагах рассмотрения матриц мы ограничимся двумя видами матриц, это матрица в виде строки или столбца, а также прямоугольная матрица. Далее, распространим рассмотренные правила и смыслы на многомерные матрицы (или, в другой терминологии, многомерные массивы).

Начнем с того, что, все матрицы MatLab являются динамическими структурами, то есть, они могут изменять свои размеры в процессе выполнения над ними операций. Динамический характер матриц обусловил одно из важнейших свойств MatLab – любая переменная в MatLab является матрицей.

Все переменные в MATLAB это матрицы

Для проверки этого утверждения воспользуемся простейшим примером:

```
% Определим переменную X и присвоим ей комплексное
% значение 1 + 2j

>> X = 1+2j

% Обратимся к переменной X как к прямоугольной
% матрице с индексом строки и индексом столбца
% равными единице

>> X(1,1)
ans =
    1.0000 + 2.0000i
```

Определение одномерных матриц (векторов).

Под одномерными матрицами будем подразумевать матрицы, состоящие из одной строки, содержащей несколько столбцов (вектор – строка) или состоящие из множества строк, где в каждой строке присутствует только один столбец (вектор столбец).

```
<Вектор > ::= < [ > < ] >
              | <Вектор-строка>
              | <Вектор-столбец>
```

```
<Вектор-строка> ::= < [ ><Список вектора-строки>< ] >
```

```
<Вектор-столбец> ::= < [ ><Список вектора-столбца>< ] >
```

Примечание. В некоторых случаях самоопределенные термины могут совпадать с элементами изображения

термов в формах Бэкуса–Наура. В таких случаях они будут записываться с помощью терм-скобок. Например, знак «квадратная скобка» будет записан как $\langle \ [\] \ \rangle$ или $\langle \ [\] \ \rangle$.

$\langle \text{Список вектора-строки} \rangle ::= \langle \text{Значение для вектора-строки} \rangle$
| $\langle \text{Список вектора-строки} \rangle , \langle \text{Значение для вектора-строки} \rangle$
| $\langle \text{Список вектора-строки} \rangle \langle \ \ \rangle \langle \text{Значение для вектора-строки} \rangle$

$\langle \text{Список вектора-столбца} \rangle ::= \langle \text{Значение для вектора-столбца} \rangle$
| $\langle \text{Список вектора-столбца} \rangle ; \langle \text{Значение для вектора-столбца} \rangle$

$\langle \text{Значение для вектора-строки} \rangle ::= \langle \text{Арифметическое значение} \rangle$
| $\langle \text{Вычисленное логическое значение} \rangle$
| $\langle \text{Имя вектора-строки} \rangle$
| $\langle \text{Строка-диапазон} \rangle$

$\langle \text{Значение для вектора-столбца} \rangle ::= \langle \text{Арифметическое значение} \rangle$
| $\langle \text{Вычисленное логическое значение} \rangle$
| $\langle \text{Имя вектора-столбца} \rangle$

$\langle \text{Арифметическое значение} \rangle ::= \langle \text{Целое со знаком} \rangle$
| $\langle \text{Действительное число} \rangle$
| $\langle \text{Комплексное число} \rangle$

Примечание. $\langle \text{Целое со знаком} \rangle$ $\langle \text{Действительное число} \rangle$ и $\langle \text{Комплексное число} \rangle$ было сделано в лабораторной работе №1

$\langle \text{Вычисленное логическое значение} \rangle ::= \text{true} \mid \text{false}$

Примечание. Формальное определение логических выражений, которые порождают $\langle \text{Вычисленное логическое значение} \rangle$ мы сделаем в последующих лабораторных работах.

$\langle \text{Строка-диапазон} \rangle ::= \langle \text{Начало} \rangle [: \langle \text{Шаг} \rangle] : \langle \text{Конец} \rangle$

$\langle \text{Начало} \rangle ::= \langle \text{Имя переменной} \rangle \mid \langle \text{Значение для диапазона} \rangle$

$\langle \text{Шаг} \rangle ::= \langle \text{Имя переменной} \rangle \mid \langle \text{Значение для диапазона} \rangle$

$\langle \text{Конец} \rangle ::= \langle \text{Имя переменной} \rangle \mid \langle \text{Значение для диапазона} \rangle$

$\langle \text{Значение для диапазона} \rangle ::= \langle \text{Целое со знаком} \rangle$
| $\langle \text{Действительное число} \rangle$

Примечание. Численные значения начала, шага и конца диапазона следует выбирать так, чтобы выполнялось:

Начало < Конец
Начало + Шаг > Начало
Кроме того, если шаг не задан, то его значение
равно единице.

И наконец:

<Имя вектора-строки> ::= <Идентификатор>

<Имя вектора-столбца> ::= <Идентификатор>

ПРИМЕРЫ:

1. Явное определение вектора-строки или вектора-столбца

```
>> % Явное определение вектора-строки (вариант 1)
>> X1=[1 2 3]
>> % Явное определение вектора-строки (вариант 2)
>> X2=[4, 5, 6]
```

```
>> % Явное определение вектора-столбца
>> Y1=[1; -2; 3; -1.12; 1.55e-2]
Y1= 1.0000
    -2.0000
     3.0000
    -1.1200
     0.0155
```

2. Определение вектора-строки с использованием имени вектора-строки

```
>> X3=[7 8 X2]
X3 = 7      8      4      5      6
```

3. Определение вектора-строки с использованием значения 1X1, вычисляемого функцией и имени вектора-строки.

```
>> X3=[sqrt(4), X1]
X3 = 2      1      2      3
```

4. Определение вектора-строки с использованием значения комплексного числа и имени вектора-строки.

```
>> X4=[1+2j, X1]
X4 = 1.0000 + 2.0000i    1.0000    2.0000    3.0000
```

5. Определение вектора-строки с использованием имени вектора-строки и вычисленного логического значения.

```
>> X5=[X1, true]
X5 = 1      2      3      1
```

6. Явное определение значений вектора-строки с помощью начального значения, шага и конечного значения.

```
>> X6=[1,2, 4:8]
X6 = 1      2      4      5      6      7      8
```

```
>> X6= [2.5 : 0.1 : 2.8]
X6= 2.5000    2.6000    2.7000    2.8000
```

7. Неявное определение значений вектора-строки с помощью начального значения, шага и конечного значения.

```
>> xb=2.5;
>> step=0.1;
>> xe=2.8;
>> X7=[xb:step:xe, 1,2]
X7 = 2.5000    2.6000    2.7000    2.8000    1.0000    2.0000
```

Мы уделили основное внимание синтаксису и способам создания векторов-строк и привели только один пример создания вектора-столбца (см. пример 1). Это обусловлено тем, что преобразовать вектор-строку в вектор-столбец достаточно просто, используя операцию транспонирования. Воспользуемся результатом последнего примера и преобразуем X7 в Y7:

8. Преобразование значений вектора-строки в значения вектора-столбца.

```
>> Y7 = X7'
Y7 = 2.5000
      2.6000
      2.7000
      2.8000
      1.0000
      2.0000
```

Определение прямоугольных матриц (2D-матриц) .

Прямоугольные матрицы можно считать основной формой структуризации данных в MatLab. Такие матрицы получают объединением (конкатенацией) векторов-строк одинакового размера или объединением векторов-столбцов, также одинакового размера. Синтаксис определения прямоугольных матриц имеет вид:

```
<2D-матрица> ::= <[><Строки 2D-матрицы><]>
                | <[><Столбцы 2D-матрицы><]>
```

```
<Строки 2D-матрицы> ::= <Вектор-строка>
                        | <Строки 2D-матрицы> ; <[><Вектор-строка><]>
```

```
<Столбцы 2D-матрицы> ::= <Вектор-столбец>
                        | <Столбцы 2D-матрицы> , <[><Вектор- столбец><]>
```

ПРИМЕРЫ

1. Явное определение прямоугольной матрицы из двух векторов-строк.

```
>> A1=[[1,2,3]; [4,5,6]]  
A1 = 1      2      3  
      4      5      6
```

2. Явное определение прямоугольной матрицы из трех векторов-строк, заданных одинаковыми диапазонами.

```
>> A2=[[1:3]; [4:6]; [8:10]]  
A2 = 1      2      3  
      4      5      6  
      8      9     10
```

3. Определение прямоугольной матрицы из трех векторов-строк, заданных именами X1 и X2.

```
>> X1=[1 2 3]  
X1= 1 2 3  
>> X2=[4, 5, 6]  
X2= 4 5 6  
A3=[X1; X2]  
A3 = 1      2      3  
      4      5      6
```

Примечание: В некоторых случаях, как правило, когда вектора представлены переменными или диапазонами допускается опускать квадратные скобки, охватывающие отдельные вектора. Однако такая возможность сокращенного синтаксиса не является общим правилом и может вызывать ошибку «Error using ==> horzcat»

4. Явное определение прямоугольной матрицы из двух векторов-столбцов.

```
A4=[[1;2;3], [4;5;6]]  
A4 = 1      4  
      2      5  
      3      6
```

Правила доступа к структурам внутри матриц

Доступ к структурам внутри вектора или прямоугольной матрицы осуществляется с помощью указания координат таких структур. Самыми простейшими структурами внутри матриц являются отдельные ячейки содержимым, которых являются значения численных или других типов. Отдельные ячейки матриц нумеруются натуральными

числами (индексами) начиная от единицы. Области внутри матриц (субматрицы) указываются диапазонами индексов. Общий синтаксис доступа к внутренним структурам векторов и 2D матриц определяется следующими правилами:

```

<Субматрица> ::= <Субматрица вектора>
                | <Субматрица 2D-матрицы>

<Субматрица вектора> ::= <Имя матрицы>(<Диапазон индексов>)

<Имя матрицы> ::= <Идентификатор>

<Диапазон индексов> ::= <Индекс>
                        | <Индекс> : <Индекс>
                        | <Индекс> : end
                        | :

<Индекс> ::= <Индекс-переменная> | <Индекс-число>

<Индекс-переменная> ::= <Имя переменной>

```

Примечание: такая переменная должна содержать только значения, которые определяются как <Индекс-число>

```

<Индекс-число> ::= <Ненулевая цифра>
                  | <Индекс-число><Ненулевая цифра>

<Ненулевая цифра> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<Субматрица 2d-матрицы> ::=
    <Имя матрицы>(<Индексы-строки>,<Индексы-колонки>)

<Индексы-строки> ::= <Диапазон индексов>
<Индексы-колонки> ::= <Диапазон индексов>

```

ПРИМЕРЫ:

- Доступ к ячейке с индесами строки=2 и столбца=4


```

A = [[1,2,3,4,5];[6,7,8,9,10];[11,12,13,14,15]]
B = [] % Обнулим матрицу B
B = A(2,4)

```
- Доступ к субматрице (области внутри матрицы) с диапазонами по строкам от 2 до 3 и столбцами от 4 до 5


```

A = [[1,2,3,4,5];[6,7,8,9,10];[11,12,13,14,15]]
B = ones(8,8)
B(1:2,1:2) = A(2:3,4:5)

```
- Доступ к субматрице (области внутри матрицы) с индесом строки=5 и диапазоном по столбцами от 3 до 5

```

A = [[1,2,3,4,5];[6,7,8,9,10];[11,12,13,14,15]]
B = [] % Обнулим матрицу B
A(3,3:5)
B(5,7:9) = A(3,3:5)

```

4. Доступ к субматрице (области внутри матрицы) с диапазонами по строкам от 2 до 3 и столбцами от 4 до последнего в матрице

```

A = [[1,2,3,4,5];[6,7,8,9,10];[11,12,13,14,15]]
B = [] % Обнулим матрицу B
B = A(2:3,4:end)

```

5. Доступ к субматрице (области внутри матрицы) с диапазонами по всем строкам и столбцами от 4 до последнего в матрице

```

A = [[1,2,3,4,5];[6,7,8,9,10];[11,12,13,14,15]]
B = [] % Обнулим матрицу B
B = A(:,4:end)

```

Матрицы MatLab это динамические структуры

Динамическими называют данные, если в процессе выполнения над ними операций они изменяют не только свои значения, но и структуру представления. Для матриц MatLab минимальными структурами, над которыми реализуются динамические преобразования, являются вектора.

Примеры динамического расширения размеров 2D-матриц.

1. Создание пустой матрицы

```

>> A=[]
A = []

```

Примечание. Такая матрица представлена только своим именем и не содержит ни одной строки и, соответственно, ни одного столбца.

2. Создание нулевой матрицы, заданного размера. Например, 3 строки, 4 столбца.

```

>> A(3,4)=0
A =  0      0      0      0
     0      0      0      0
     0      0      0      0

```

Примечание. Аналогичную операцию выполняет стандартная в MatLab функция `zeros(3,4)`.

3. Расширение существующей матрицы до заданного размера.

- 3.1. Пусть в процессе вычислений у нас получилась некоторая матрица, которую мы сформируем следующим образом:

```

A=[1:3; 4:6; 7:9]
A =  1      2      3
     4      5      6

```

7 8 9

3.2. Если у нас возникла необходимость ее расширить, то достаточно задать всего одну ячейку за пределами ее текущей размерности, например:

```
>> A(4,2)=42
A =   1       2       3
      4       5       6
      7       8       9
      0      42       0
```

или, если необходимо выполнить еще более радикальное расширение, то:

```
>> A(6,7)=67
A =   1       2       3       0       0       0       0
      4       5       6       0       0       0       0
      7       8       9       0       0       0       0
      0      42       0       0       0       0       0
      0       0       0       0       0       0       0
      0       0       0       0       0       0      67
```

Примечание. Конкретные значения, которые мы присваивали ячейке, расширяющей матрицу, выбраны только из соображения наглядности и могут быть произвольными в рамках допустимых типов.

Примеры манипуляции строками и столбцами или субматрицами в 2D-матрицах.

4. Достаточно часто необходимо возникает задача изменить значения строки, столбца или субматрицы в рамках некоторой исходной матрицы. Проиллюстрируем решение этой задачи на примере перемещения двух субматриц внутри исходной матрицы без потери их значений.

4.1. Создаем из строк исходную матрицу по сокращенному синтаксису

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

4.2. Выполняем сохранение значений первой субматрицы в буферной (временной) переменной TEMP.

```
>> TEMP=A(1:2, 1:2)
TEMP =   1       2
        5       6
```


4.3. Копируем вторую субматрицу на место первой

```
>> A(1:2, 1:2)=A(2:3, 3:4)
A = 7      8      3      4
     11     12     7      8
          9     10     11     12
```

Примечание. Размерности и размеры субматриц слева и справа от операции присвоения должны быть строго одинаковы.

4.4. Перемещаем первую (сохраненную) субматрицу на место второй

```
>> A(2:3, 3:4)=TEMP
A = 7      8      3      4
     11     12     1      2
          9     10     5      6
```

4.5. Динамически освобождаем временную переменную TEMP, то есть, удаляем ее из Workspace.

```
>> clear TEMP
```

Примечание. Функция `clear` в системе MatLab позволяет динамически удалять из Workspace все, несколько или конкретную переменную. Синтаксис функции `clear` смотрите в HELP.

Примеры динамического понижения размеров исходной 2D-матрицы.

5. Также достаточно часто (например, при построении миноров), необходимо удалить некоторую строку или столбец, что равносильно динамическому изменению размеров исходной матрицы. Покажем способ реализации такой операции

5.1. Создаем из строк исходную матрицу.

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

5.2. Удалим в матрице второй столбец.

```
>> A(:,2)=[ ]
A = 1      3      4
     5      7      8
     9     11     12
```

5.3. Далее, удалим вторую строку:

```
>> A(2,:)=[]  
A = 1     3     4  
     9    11    12
```

Матрицы с размерностью больше чем 2D

В материале, который мы рассматривали ранее, размерность матриц не превышала значения 2. Например, размерность векторов равна единице или 1D(one-dimensional array), размерность прямоугольных матриц равна двум или 2D (two-dimensional array). В отличие от термина «размер» (size), который определяет количество ячеек, термин «размерность» определяет необходимое число индексов (или координат) для однозначного обращения к одной ячейки матрицы.

Система MatLab позволяет работать с матрицами размерность которых может превышать 2D. Такие матрицы называют многомерными матрицами(multi-dimensional array). Создание таких матриц синтаксически подобно созданию 2D-матриц.

Пример создания матриц 3D

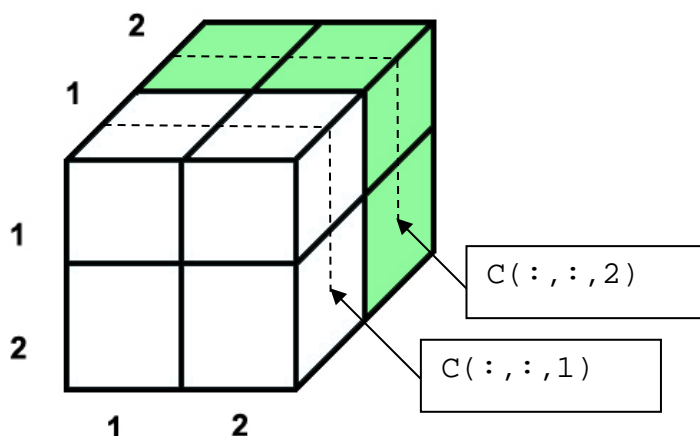
В качестве примера покажем создание матрицы 3D как кубика с размерами две ячейки в каждой размерности:

```
C(2,2,2)=222
```

Результат:

```
C(:, :, 1) = 0     0  
             0     0  
C(:, :, 2) = 0     0  
             0    222
```

Результат этой операции явно требует пояснений. Итак, представим заданную матрицу в виде следующего рисунка:



Тогда результат, обозначенный как $C(:, :, 1)$, отобразит нам все значения ячеек ее первого слоя, а результат, обозначенный как $C(:, :, 2)$, все значения ячеек ее второго слоя

Примечание. Такой способ создания можно назвать как создание матриц путем указания конечных индексов на главной диагонали.

Пример манипуляции значениями и размерностью матриц 3D

В 3D-матрице можно манипулировать не только значениями конкретных ячеек или векторов как 2D-матрице, но также можно изменять целые слои. Для этого конкретный слой необходимо зафиксировать в одной из размерностей, после чего заменить значения слоя.

1. Манипуляция значениями матрицы, в которой слои фиксируется по **третьей** размерности:

```
% Сформируем исходную матрицу
>> C(2,2,2)=222;
```

```
% Подготовим значения заменяющие значения слоя
>> V=[1,2; 3,4];
```

```
% Заменяем матрицей «V» первый слой матрицы «C»
>> C(:, :, 1)=V
C(:, :, 1) = 1      2
              3      4
C(:, :, 2) = 0      0
              0     222
```

```
% Заменяем первую строку (вектор) во втором слое
>> C(1, :, 2)=[5, 6]
C(:, :, 1) = 1      2
              3      4
C(:, :, 2) = 5      6
              0     222
```

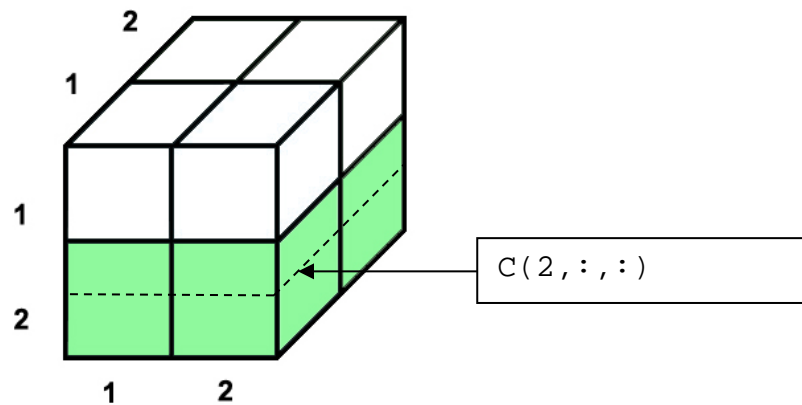
```
% В завершение удалим второй слой полностью
>> C(:, :, 2)=[]
C = 1      2
     3      4
```

Примечание. Подчеркнем, что последняя операция в этом примере изменяет не только значения матрицы но и ее размерность.

2. Манипуляция значениями матрицы, в которой слои фиксируется по **первой** размерности:

```
% Сформируем исходную матрицу
>> C(2,2,2)=222;
```

% Покажем фиксацию второго слоя по первой размерности с помощью следующего рисунка:



% Заменяем (явно заданной) 2D-матрицей зафиксированный слой матрицы «С»

```
>> C(2, :, :) = [[1, 2]; [3, 4]]
C(:, :, 1) = 0      0
              1      3
C(:, :, 2) = 0      0
              2      4
```

Примечание. Обратите внимание – не смотря на то, что операция осуществлялась в слое с фиксированной строкой, результат отображается слоями (сечениями) с фиксацией по третьей размерности. Эргономика такого отображения это достаточно субъективная территория или как шутят программисты – это чисто религиозный вопрос ☺

Несколько слов о матрицах размерностью 4D и выше

Я думаю, Вы уже догадались, что индексы четвертой размерности позволяют создавать и избирательно работать с множеством 3D-матриц. Такое правило работает и для более высоких размерностей.

К сожалению, с повышением размерности отображение результата становится все менее и менее читаемым. В этом легко убедиться, выполнив создание, к примеру, 8D-матрицы:

```
>> A(2,2,2,2,2,2,2,2)=0
```

Отображение результата такой операции представляет собой 64 сечения в виде 2D-матрицы каждое. Это создает заметные трудности при планировании обработки значений, особенно при работе со средними (в смысле размерностей) структурами внутри таких матриц. Одним из выходов в такой ситуации можно считать переход к программным средствам обработки, которые мы рассмотрим в дальнейшем.

Сервисные функции MatLab для работы с матрицами

Большинство функций MatLab имеют много вариантов списка их параметров, а соответственно и смысла каждого из них. Детальное описание таких параметров для каждой функции явно выходит за рамки лабораторной работы. По этой причине, мы ограничимся только именами функций и кратким смыслом их действий. Для получения полной информации обращайтесь в подсистему HELP.

1. Группа функций, которая позволяет получить размеры динамически изменяющихся матриц

№	Имя функции	Описание
1	length	Возвращает длину вектора в ячейках
2	size	Возвращает для каждой или выбранной размерности матрицы длину этой размерности в ячейках

2. Группа функций, которая позволяет создавать и инициализировать типичные варианты матриц.

№	Имя функции	Описание
1	zeros	Создает матрицу заданной размерности, ячейки которой заполнены нулями.
2	ones	Создает матрицу заданной размерности, ячейки которой заполнены единицами.
3	eye	Создает матрицу заданной размерности, ячейки главной диагонали которой, заполнены единицами, а остальные ячейки заполнены нулями.
4	rand	Создает матрицу заданной размерности, ячейки которой заполнены случайными значениями в диапазоне (0-1) в соответствии с равномерным законом распределения.
5	randn	Создает матрицу заданной размерности, ячейки которой заполнены случайными значениями в диапазоне (0-1) в соответствии с нормальным законом распределения.

3. Группа функций для выполнения типичных трансформаций в матрицах.

№	Имя функции	Описание
1	fliplr	Выполняет для векторов и 2D-матриц зеркальное отображение относительно вертикали.
2	flipud	Выполняет для векторов и 2D-матриц зеркальное отображение относительно горизонтали.
3	rot90	Выполняет для квадратных 2D-матриц вращение значений матрицы на углы кратные 90°.

Примечание. В приведенных таблицах представлены только наиболее часто используемые функции, условно разделенные на три категории. На самом деле, таких функций, особенно в категориях 2 и 3 значительно больше. Подсистема HELP с помощью вкладки SEARCH и гиперссылок позволяет подобрать необходимую Вам функцию из множества функций MatLab.

ПОЭЛЕМЕНТНЫЕ ОПЕРАЦИИ С МАТРИЦАМИ В MATLAB.

Операции MATLAB над матрицами подразделяются на два класса операций – это поэлементные операции и матричные операции.

Поэлементные операции применимы к матрицам любой размерности. Главное при двухместных поэлементных операциях это использование одинаковых размеров в матрицах операндах.

Матричные операции, как правило, применяются для решения задач линейной алгебры и в данной лабораторной работе не рассматриваются. Рассмотрение таких операций будет выполняться по мере необходимости их использования в различных дисциплинах учебного процесса. В рамках данной дисциплины такие операции будут рассмотрены в дополнительных лабораторных работах, ориентированных на самостоятельное изучение.

В рамках данной работы рассмотрим только одноместную матричную операцию – <транспонирование>, которая определена для векторов и прямоугольных матриц.

<транспонированная матрица> ::= <матрица>'

Данная операция выполняет перестановку местами строк и столбцов исходной матрицы, то есть все строки исходной матрицы становятся столбцами результирующей матрицы.

Например:

```
% Пример транспонирования вектора - строки
A=[1 2 3 4]
B=A'
```

Результат:

```
A =
    1     2     3     4
B =
    1
    2
    3
    4
```

```
% Пример транспонирования вектора - столбца
A=[1; 2; 3; 4]
```

B=A'

Результат:

A =

1
2
3
4

B =

1 2 3 4

% Пример транспонирования прямоугольной матрицы

A=[1:3; 5:7; 9:11]

B=A'

Результат:

A =

1 2 3
5 6 7
9 10 11

B =

1 5 9
2 6 10
3 7 11

Итак, ранее мы сформулировали обобщенное правило для нетерминала <Выражение>

```
<Выражение> ::= <Простое выражение>
               | <Одноместная операция><Простое выражение>
               | <Выражение><Двухместная операция><Простое выражение>
               | ( <Выражение> )
```

```
<Простое выражение> ::= <Операнд>
                       | <Одноместная операция><Операнд>
                       | <Операнд><'>
                       | <Операнд><Двухместная операция><Операнд>
```

```
<Операнд> ::= <Константа>
              | <Матрица>
              | <Вызов функции>
```

```
<Вызов функции> ::= <Имя функции>
                   | <Имя функции> ( )
                   | <Имя функции> ( <Список параметров> )
```

```
<Имя функции> ::= <Идентификатор>
```

```
<Список параметров> ::= <Параметр>
                       | <Параметр> , <Список параметров>
```

Как уже говорилось, приретенная формализация не раскрывает определения нетерминалов <Одноместная операция> и <Двухместная

операция>. Кроме того подчеркивалось, что операции в MatLab обладают весьма высоким полиморфизмом, то есть, смысл их выполнения и даже состав существенно зависит от типа операндов.

В рамках данной лабораторной работы мы уделим основное внимание поэлементным арифметическим операциям, операндами которых являются матрицы различной размерности.

Одноместные поэлементные операции для арифметических матриц любой размерности

Для арифметических матриц любой размерности определяется единственная одноместная операция (минус):

<Одноместная операция для nD матриц> ::= -

Практически, это означает изменение знака на противоположный для всех значений в ячейках матрицы, например:

Пусть:

```
>>A
A(:, :, 1) = 2      2
             2      2
A(:, :, 2) = 2      2
             2      2
```

Тогда:

```
>> -A
ans(:, :, 1) = -2      -2
              -2      -2
ans(:, :, 2) = -2      -2
              -2      -2
```

Двухместные поэлементные операции для арифметических nD-матриц и простого арифметического операнда (скаляра).

Напомним, что простой арифметический операнд мы уже рассматривали в лабораторной работе №1 в виде:

<Простой арифметический операнд> ::= <Арифметическая константа>
| <Имя переменной (размерности 1x1)>
| <Вызов функции (с результатом размерности 1x1)>

При этом под размерностью 1x1 мы подразумевали матрицу с единственной ячейкой. Следует также отметить, что простой арифметический операнд часто называют скаляром.

Для такого сочетания операндов двухместная арифметическая операция определяется правилом:

<двухместная арифметическая операция для nD-матрицы и скаляра> ::= - | + | * | / | ^ | ./ | .* | ./ | .^

Обратим внимание, что помимо символов традиционных операций умножения, деления и возведения в степень (* | / | ^), используются их двухсимвольные обозначения с лидирующей точкой. Такие двухсимвольные обозначения подчеркивают необходимость выполнять операцию поэлементно, то есть, выполнять соответственно для каждой ячейки nD-матрицы и ячейки простого арифметического операнда (**скаляра**). Необходимо подчеркнуть, что для операций между скаляром и скаляром, а также матрицей и скаляром операции (* | / | ^) и (.* | ./ | .^) выполняются совершенно одинаково. Очевидно, что конечным результатом такой операции будет либо скаляр, либо nD-матрица.

Например:

1. Пусть имеются исходная кубическая матрица A. Для начала выполним ее создание:

```
A = ones(3,3,3)
A * 4
```

Результат:

```
ans(:,:,1) = 4      4
              4      4
              4      4
ans(:,:,2) = 4      4
              4      4
              4      4
```

2. На базе полученных в (1) значений матрицы A приведем еще один пример, который иллюстрирует возможные позиции операндов в рассматриваемой операции:

```
>> 4 / A
```

Результат:

```
ans(:,:,1) = 1      1
              1      1
              1      1
ans(:,:,2) = 1      1
              1      1
              1      1
```

или (напомним что $\text{sqrt}(4) = 2$)

```
(A.*2)./sqrt(4)
```

Результат:

```
ans(:,:,1) = 1      1
              1      1
```

```

      1      1
ans(:,:,2) = 1      1
      1      1
      1      1

```

3. Еще один важный пример показывает, что комплексное число интерпретируется в MatLab как арифметическое выражение, состоящее из суммы действительной и мнимой частей:

```

A = ones(3,3,3).*2
A .^ (2 + 2i)

```

Результат:

```

ans(:,:,1) = 0.7338 + 3.9321i    0.7338 + 3.9321i
              0.7338 + 3.9321i    0.7338 + 3.9321i
              0.7338 + 3.9321i    0.7338 + 3.9321i

ans(:,:,2) = 0.7338 + 3.9321i    0.7338 + 3.9321i
              0.7338 + 3.9321i    0.7338 + 3.9321i
              0.7338 + 3.9321i    0.7338 + 3.9321i

ans(:,:,3) = 0.7338 + 3.9321i    0.7338 + 3.9321i
              0.7338 + 3.9321i    0.7338 + 3.9321i
              0.7338 + 3.9321i    0.7338 + 3.9321i

```

Однако, учитывая старшинство операций при выполнении следующих строк, мы уже получим такой результат:

```

A = ones(3,3,3).*2
A .^ 2 + 2i

```

Результат:

```

ans(:,:,1) = 4.0000 + 2.0000i    4.0000 + 2.0000i
              4.0000 + 2.0000i    4.0000 + 2.0000i
              4.0000 + 2.0000i    4.0000 + 2.0000i

ans(:,:,2) = 4.0000 + 2.0000i    4.0000 + 2.0000i
              4.0000 + 2.0000i    4.0000 + 2.0000i
              4.0000 + 2.0000i    4.0000 + 2.0000i

```

Двухместные поэлементные операции для двух арифметических nD-матриц одинаковой размерности и размера.

Если операндами двухместной операции являются nD- матрицы одинаковой размерности и размера то результатом операции является является nD-матрица аналогичной размерности и размера, а значения ее ячеек будут вычисляться как результат приенения операции к значениям ячеек nD-матриц операндов с одинаковыми индексами. Математически это можно записать следующим образом:

Для каждого полностью определенного варианта значений индексов nD-матриц:

$inx1, inx2, \dots, inxN$

поэлементно выполнить:

$Y (inx1, \dots, inxN) = X1 (inx1, \dots, inxN) < \text{оператор} > X2 (inx1, \dots, inxN)$

Назовем такую операцию nD-двухместной арифметической операцией.

<nD-двухместная арифметическая операция > ::=

- | + | .* | ./ | .^

Как видно из приведенного правила, операции * и / исключены из списка возможных. Это связано с тем, что исключенные символы умножения и деления используются в MatLab не только для поэлементных, но и для матричных операций, которые определяются в линейной алгебре.

Приведем несколько примеров двухместных операций для двух арифметических nD-матриц одинаковой размерности и размера.

1. Пусть имеются две исходные кубические матрицы A и B одинаковой размерности. Для начала выполним их создание:

```
A = ones(3,3,3);
B = ones(3,3,3).*2;
```

2. Пример поэлементной операции деления:

```
C = A ./ B
```

Результат:

```
C(:, :, 1) = 0.5000    0.5000
             0.5000    0.5000
             0.5000    0.5000
C(:, :, 2) = 0.5000    0.5000
             0.5000    0.5000
             0.5000    0.5000
C(:, :, 3) = 0.5000    0.5000
             0.5000    0.5000
             0.5000    0.5000
```

3. Пример, в котором значения матрицы C используются как набор показателей степени для каждого элемента матрицы B:

```
B .^ C
```

Результат:

```
ans(:, :, 1) = 1.4142    1.4142
               1.4142    1.4142
               1.4142    1.4142
ans(:, :, 2) = 1.4142    1.4142
               1.4142    1.4142
```

```
ans(:, :, 3) =    1.4142    1.4142
                 1.4142    1.4142
                 1.4142    1.4142
                 1.4142    1.4142
```

ПРАКТИЧЕСКАЯ РАБОТА.

1. Выполните все примеры из теоретической части лабораторной работы.
2. Опробуйте создание и трансформации матриц с помощью типичных функций и для различных вариантов списка их параметров.

Финальная редакция материала 23.08.2018г.
Воронов С.И.