ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Вычислительные системы и информационная безопасность»

Учебное пособие

«Основы работы с массивами в пакете Matlab»

Автор
Цветкова О.Л.

Ростов-на-Дону, 2018

Аннотация

«Учебное пособие» предназначено для студентов очной формы обучения направления 10.03.01 Информационная безопасность, заочной формы обучения направления 09.03.02 Информационные системы и технологии.

Авторы

доцент, к.т.н,
доцент кафедры «Вычислительные системы и информационная безопасность»
Цветкова О.Л.

Оглавление

1. Двумерные массивы (матрицы)

Массив — это упорядоченная, пронумерованная совокупность однородных данных, имеющая имя. Массивы различаются по числу размерностей (измерений): одномерные, двумерные, многомерные.

В русской литературе понятия «размер» и «размерность» массивов являются почти синонимами. Однако в литературе по системе Matlab они имеют разный смысл. Под размерностью массивов понимается число измерений в пространственном представлении массивов, а под размером — число строк и столбцов (m×n) в каждой размерности массива.

В Matlab матрица — это прямоугольный массив чисел. Особое значение придается матрицам 1×1, которые являются скалярами, и матрицам, имеющим один столбец или одну строку, — векторам. Вектор и матрица являются математическими объектами, а одномерные, двумерные или многомерные массивы — способы хранения этих объектов в компьютере.

Все данные Matlab представляет в виде массивов. Таким образом, по умолчанию Matlab предполагает, что каждая заданная переменная — это вектор, матрица или массив. Все определяется конкретным значением переменной. Например, если задано X=1, то это значит, что X — это матрица размера 1×1.

Ввод матриц. Матрицы в Matlab можно вводить несколькими способами:

— вводить полный список элементов;

— генерировать матрицы, используя встроенные функции;

— загружать матрицы из внешних файлов.

1. Ввод матриц списком элементов. Необходимо следовать условиям:

— отделять элементы строки пробелами или запятыми;

— использовать точку с запятой ; для обозначения окончания каждой строки;

— окружать весь список элементов оператором объединения [ ].

Например:

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

Matlab отобразит матрицу:

```
A =
    16  3   2  13
     5 10  11   8
```

```
 9  6     7     12
 4 15    14      1
```

Возможен ввод элементов матриц и векторов в виде арифметических выражений, содержащих любые доступные системе функции, например:

```
V= [2+2/(3+4),exp(5),sqrt(10)];
V =
  2.2857 148.4132 3.1623
```

Возможно задание векторов и матриц с комплексными элементами вида a + b * i, где a — действительная часть числа, b — мнимая часть и i — мнимая единица (корень квадратный из –1), например выражения:

```
CM = [1 2; 3 4] + i*[5 6; 7 8]
```

и

```
CM = [1+5*i 2+6*i; 3+7*i 4+8*i]
```

дадут одинаковый результат:

```
CM =
  1.0000 + 5.0000i 2.0000 + 6.0000i
  3.0000 + 7.0000i 4.0000 + 8.0000i
```

2. Генерирование матриц с использованием встроенных функций. Matlab имеет функции, которые создают основные матрицы:

zeros(m, n) — матрица размера $m \times n$ с нулевыми элементами;

ones(m, n) — матрица размера $m \times n$ с единичными элементами;

eye(m, n) — единичная матрица размера $m \times n$;

rand(m, n) — матрица размера $m \times n$ со случайными элементами, имеющими равномерное распределение в промежутке (0, 1);

randn(m, n) — матрица размера $m \times n$ со случайными элементами, распределенными по нормальному закону с нулевым математическим ожиданием и среднеквадратическим отклонением, равным 1.

Пример.

```
Z = zeros(2,4)

Z =
     0     0     0     0
     0     0     0     0

F = 5*ones(3,3)

F =
     5     5     5
     5     5     5
     5     5     5

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
```

3. Загрузка матриц из внешних файлов. 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.

Пример. Пусть текстовый файл в Блокноте:

```
16.0    3.0    2.0   13.0
 5.0   10.0   11.0    8.0
 9.0    6.0    7.0   12.0
 4.0   15.0   14.0    1.0
```

сохранен по именем magik.dat.

Тогда команда:

load magik.dat

прочитает этот файл и создаст переменную magik, содержащую матрицу.

Обращение к элементам матриц. Элемент в строке i и столбце j матрицы A обозначается A(i,j). Например, A(4,2) — это число в четвертой строке и втором столбце. Для матрицы A = [16

3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]:

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

Таким образом, можно вычислить сумму элементов в четвертом столбце матрицы A:

```
A(1,4) + A(2,4) + A(3,4) + A(4,4)
ответ
ans =
  34
```

Однако это не самый лучший способ суммирования отдельной строки. Для этой цели больше подходит оператор двоеточие :. Индексное выражение, включая двоеточие, относится к части матрицы: A(1:k, j) — это первые k элементов j-го столбца матрицы A. Так:

```
sum(A(1:4,4))
```

вычисляет сумму элементов четвертой строки.

Но есть и лучший способ. Двоеточие, само по себе, обращается ко всем элементам в строке или столбце матрицы, а слово end — к последней строке или столбцу. Так:

```
sum(A(:,end))
```

вычисляет сумму всех элементов в последнем столбце матрицы A:

```
ans =
  34
```

Также возможно обращаться к элементам матрицы через один индекс, A(k). В этом случае массив рассматривается как длинный вектор, сформированный из столбцов исходной матрицы.

Так, A(8) — это другой способ ссылаться на значение 15, хранящееся в A(4,2).

Если использовать значение элемента вне матрицы, Matlab выдаст ошибку:

```
t=A(4,5)
??? Index exceeds matrix dimensions.
```

С другой стороны, если сохранить значение вне матрицы, то размер матрицы увеличивается:

```
X=A;
X(4,5) = 17
```

```
X =
    16  3     2    13     0
    510    11     8     0
     9  6     7    12     0
     415    14     1    17
```

Удаление столбцов и строк матриц. Для формирования матриц и выполнения ряда матричных операций возникает необходимость удаления отдельных столбцов и строк матрицы. Для этого используются пустые квадратные скобки — [ ].

Пример. Пусть имеется матрица M:

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

Удалить второй столбец, используя оператор : (двоеточие) можно следующим образом:

```
M(:,2)=[ ]
M =
 1 3
 4 6
 7 9
```

Удалить вторую строку:

```
M(2,:)=[ ]
M =
 1 3
 7 9
```

Объединение (конкатенация) матриц. Объединение (конкатенация) — это процесс соединения маленьких матриц для создания больших.

Пример. Для матрицы A:

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

команда:

```
B = [A A+32; A+48 A+16]
```

даст в качестве результата матрицу размера 8×8, которая представляет собой соединение четырех подматриц:

```
B = 16 3 2 13 48 35 34 45
  5 10 11 8 37 42 43 40
  9 6 7 12 41 38 39 44
  4 15 14 1 36 47 46 33
  64 51 50 61 32 19 18 29
  53 58 59 56 21 26 27 24
```

57 54 55 60 25 22 23 28
52 63 62 49 20 31 30 17

Математические операции с матрицами. Математические операции, определенные на матрицах являются объектами линейной алгебры (Матричные функции линейной алгебры):

— det(A) возвращает определитель квадратной матрицы A;

— rank(A) возвращает ранг матрицы A;

— trace(A) возвращает след матрицы A;

— inv(A) возвращает матрицу, обратную квадратной матрицы A;

— A’ возвращает транспонированную матрицу;

— size(A) возвращает вектор-строку, содержащую количество строк и столбцов в массиве A;

— length(X) возвращает длину вектора X;

— max(A) возвращает вектор-строку, содержащую значения максимальных элементов в столбцах матрицы A;

— min(A) возвращает вектор-строку, содержащую значения минимальных элементов в столбцах матрицы A.

Поэлементные операции с массивами. Арифметические операции на массивах производятся поэлементно. Это означает, что суммирование и вычитание являются одинаковыми операциями для матриц и массивов, а например, умножение для них различно. Matlab использует точку, как часть записи, для поэлементных операций над массивами:

+ суммирование
- вычитание
.* поэлементное умножение
./ поэлементное деление
.\ поэлементное левое деление
.^ поэлементное возведение в степень

Пример. Если матрицу A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]:

```
A =
 16 3 2 13
 5 10 11 8
 9 6 7 12
 4 15 14 1
```

умножить на себя по правилам умножения массивов:

```
A.*A
```

результатом будет массив, содержащий квадраты элементов исходного массива:

```
ans =
        256   9         4       169
         25100       121        64
         81 36        49       144
         16225       196         1
```

Операции над массивами полезны для создания таблиц. Пусть n — это вектор-столбец:

```
n = (0:9)';
```

Тогда:

```
pows = [n n.^2 2.^n]
```

создает таблицу квадратов и степеней двойки:

```
pows =
 0 0 1
 1 1 2
 2 4 4
 3 9 8
 4 16 16
 5 25 32
 6 36 64
 7 49 128
 8 64 256
 9 81 512
```

Элементарные математические функции работают с массивами поэлементно. Так:

```
x
 =
(1:0.
1:2)'
```

```
s
 =  [x
sin(x)
]
```

создает таблицу значений синуса:

```
s =
 1.0000 0.8415
 1.1000 0.8912
 1.2000 0.9320
 1.3000 0.9636
 1.4000 0.9854
 1.5000 0.9975
 1.6000 0.9996
 1.7000 0.9917
```

1.8000 0.9738
1.9000 0.9463
2.0000 0.9093

2. Многомерные массивы

Понятие о многомерных массивах. Многомерные массивы характеризуются размерностью более двух. Таким массивам можно дать наглядную интерпретацию. Так, матрицу (двумерный массив) можно записать на одном листе бумаги в виде строк (rows) и столбцов (columns), состоящих из элементов матрицы, — рис. 1. Тогда блокнот с такими листами можно считать трехмерным массивом (рис. 2), полку в шкафу с блокнотами — четырехмерным массивом, шкаф со множеством полок — пятимерным массивом и т.д.

С многомерными массивами могут выполняться те же операции и вычисления, что и с двумерными массивами (матрицами). В частности, это относится ко всем операциям, осуществляемым поэлементно, а также к функции sum, mean, cross и др.

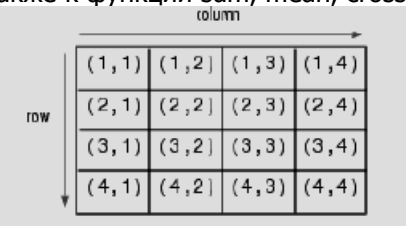

Рис. 1. Представление двумерного массива (матрицы)

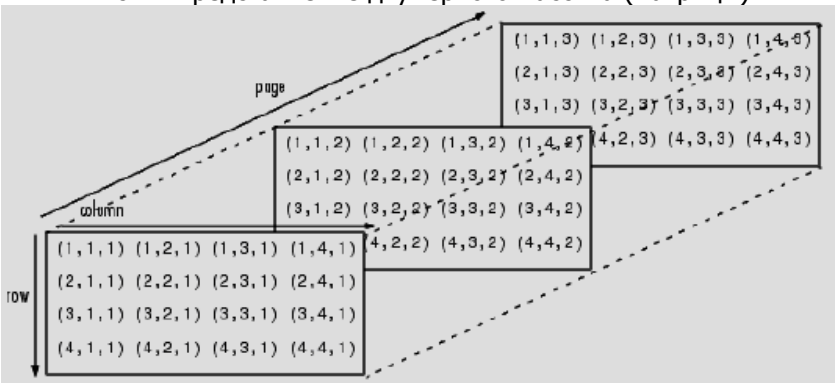

Рис. 2. Представление трехмерного массива, содержащего ряд страниц

Применение оператора : в многомерных массивах. Для выделения отдельных страниц многомерных массивов можно использовать оператор :. Подобные операции наглядно иллюстри-

рует рис. 3.

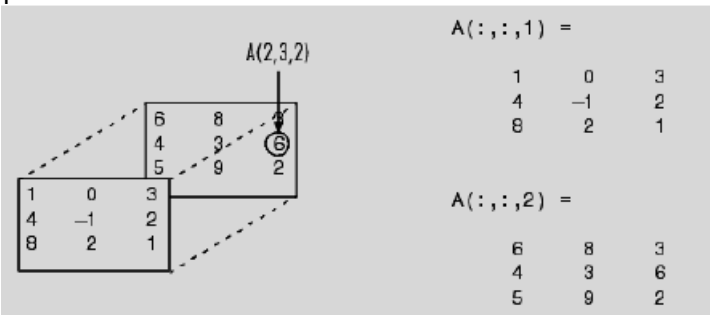

Рис. 3. Примеры работы с трехмерным массивом

Пример. Пусть задан исходный двумерный массив M размера 3×2:

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

Для добавления новой страницы с тем же размером можно расширить M следующим образом:

```
M(:,:,2)=[10 11 12; 13 14 15; 16 17 18]
M(:,:,1) =
 1 2 3
 4 5 6
 7 8 9
M(:,:,2) =
 10 11 12
 13 14 15
 16 17 18
```

Удаление размерности у многомерного массива. Удалить отдельные столбцы (строки) можно путем присвоения им значений пустого вектора-столбца [ ] (вектора-строки). Этот прием нетрудно распространить на страницы и вообще размерности многомерного массива.

Пример. Удаление первой страницы полученного массива M:

```
M(:,:,1)=[ ]
M =
 10 11 12
 13 14 15
 16 17 18
```

Доступ к отдельному элементу многомерного масси-

ва. В многомерных массивах используется то же правило индексации, что и в одномерных и двумерных. Произвольный элемент, например, трехмерного массива, задается как M(i,j,k), где i — номер строки, j — номер столбца и k — номер страницы. Этот элемент можно вывести, а можно присвоить ему заданное значение x: M(i,j,k)=x.

Создание страниц, заполненных константами и случайными числами. Если после знака присваивания стоит численная константа, то соответствующая часть массива будет содержать элементы, содержащие данную константу.

Пример.

```
M(:,:,2)=1
M(:,:,1) =
   10 11 12
   13 14 15
   16 17 18
M(:,:,2) =
    1 1 1
    1 1 1
    1 1 1
```

Замена первой страницы массива на страницу с нулевыми элементами:

```
M(:,:,1)=0
M(:,:,1) =
 0 0 0
 0 0 0
 0 0 0
M(:,:,2) =
 1 1 1
 1 1 1
 1 1 1
```

Функции ones, zeros, rand и randn. Функции ones (создание массивов с единичными элементами), zeros (создание массивов с нулевыми элементами) и rand или randn (создание массивов с элементами — случайными числами с равномерным и нормальным распределением) могут также использоваться для создания многомерных массивов.

Пример.

```
E=ones(3,3,2)
E(:,:,1) =
 1 1 1
 1 1 1
```

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

Z=zeros(2,2,3)
Z(:,:,1) =
 0 0
 0 0
Z(:,:,2) =
 0 0
 0 0
Z(:,:,3) =
 0 0
 0 0

R=randn(3,2,2)
R(:,:,1) =
 -0.4326 0.2877
 -1.6656 -1.1465
 0.1253 1.1909
R(:,:,2) =
 1.1892 0.1746
 -0.0376 -0.1867
 0.3273 0.7258
```

Вычисление числа размерностей массива. Функция ndims(A) возвращает размерность массива A:

```
M=rand(2,3,4,5);
ndims(M)
ans =
  4
```

3. Структуры и массивы структур

Структура содержит разнородные данные, относящиеся к некоторому именованному объекту.

Многомерные массивы и массивы структур находят широкое применение. Например, многомерные массивы используются для представления цветных изображений формата RGB. Они состоят из массивов интенсивности трех цветов — красного r, зеленого g и синего b.

Более сложные массивы структур нужны для разработки

баз данных, например о работниках предприятия, службах города, городах страны и т.д.

Создание структур и доступ к их компонентам. Для создания структур можно использовать операторы присваивания:

Имя_структуры.Имя_поля = значение

Пример.

```
man.name='Иван';
man.surname='Иванов';
man.date=1956;
man.height=170.5;
man.weight=70.34;
```

Теперь можно просмотреть полученную структуру, просто указав ее имя:

```
man =
 name: 'Иван'
 surname: 'Иванов'
 date: 1956
 height: 170.5000
 weight: 70.3400
```

Компоненты структуры можно вызывать по имени и менять их значения. При этом имя компонента состоит из имени структуры и имени поля, разделенных точкой.

Пример.

```
man.date=1964
man =
 name: 'Иван'
 surname: 'Иванов'
 date: 1964
 height: 170.5000
 weight: 70.3400
```

Для создания массива структур вводится их индексация. Например, вектор структур можно создать, введя индекс в скобках после имени структуры.

Пример. Создание новой, второй структуры:

```
man(2).name='Петр';
man(2).surname='Сидоров';
man(2).date=1959;
man(2)

ans =
 name: 'Петр'
 surname: 'Сидоров'
```

```
date: 1959
height: [ ]
weight: [ ]
```

В этом примере не все поля данной структуры заполнены. Поэтому значением двух последних компонентов структуры 2 оказываются пустые массивы.

Создание структур с помощью функции struct. Для создания структур предназначена функция:

struct('field1', VALUES1, 'field2', VALUES2,…).

Пример.

```
S=struct('student', 'Иванов', 'grup', 2, 'estimate', 'good')
S =
 student: 'Иванов'
 grup: 2
 estimate: 'good'
```

Функция возврата имен полей структуры. Следующая функция позволяет вывести имена полей заданной структуры:

fieldnames(S)

возвращает имена полей структуры S в виде массива ячеек.

Пример.

```
fieldnames(man)
ans =
 'name'
 'surname'
 'date'
 'height'
 'weight'
```

Функция возврата содержимого полей структуры. В конечном счете, работа со структурами сводится к выводу и использованию содержимого полей. Для возврата содержимого поля структуры S служит функция getfield:

getfield(S, 'field')

возвращает содержимое поля структуры S, что эквивалентно S.field.

Пример.

```
getfield(man(2),'name')
ans =
  Петр
```

Функция присваивания значений полям структуры. Для присваивания полям заданных значений используется следующая функция:

setfield(S, 'field', V)

возвращает структуру S с присвоением полю 'field' значения V, что эквивалентно S.field=V.

Пример.

```
setfield(man(2), 'name', 'Николай')
ans =
 name: 'Николай'
 surname: 'Сидоров'
 date: 1959
 height: [ ]
 weight: [ ]
```

Функция удаления полей структуры. Для удаления полей структуры можно использовать следующую функцию:

— rmfield(S, 'field') возвращает структуру S с удаленным полем;

— rmfield(S, FIELDS) возвращает структуру S с несколькими удаленными полями. Список удаляемых полей FIELDS задается в виде массива символов или строкового массива ячеек.

Пример.

```
rmfield(man(2), 'surname')
ans =
 name: 'Петр'
 date: 1959
 height: [ ]
 weight: [ ]
```

Функция возврата длины массива структур. Число структур в массиве структур S позволяет найти функция length(S).

4. Массивы ячеек

Массив ячеек — это массив, элементами которого являются ячейки, содержащие любые типы массивов, включая массивы ячеек.

Создание массивов ячеек. Создавать массивы ячеек можно с помощью оператора присваивания. При этом индексы элементов массива заключаются в фигурные скобки { }:

```
A{1,1}='Курить вредно!';
A{1,2}=[1 2;3 4];
A{2,1}=2+3i;
A{2,2}=0:0.1:1;
```

Вывод массива ячеек в командном режиме:

```
A
A =
 'Курить вредно!' [2x2 double]
```

```
[2.0000 + 3.0000i] [1x11 double]
```

Обращение к содержимому конкретной ячейки выполняется аналогично обычным массивам, но используются не круглые, а фигурные скобки:

```
A{1,1}
ans =
  Курить вредно!

A{1,2}
ans =
 1 2
 3 4

A{2,1}
ans =
  2.0000 + 3.0000i

A{2,2}
ans =
 Columns 1 through 7
 0 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000
 Columns 8 through 11
 0.7000 0.8000 0.9000 1.0000
```

Создание ячеек с помощью функции cell. Для создания массива ячеек может использоваться функция cell:

— cell(N) создает массив ячеек из N×N пустых матриц;

— cell(M,N) или cell([M N]) создает массив ячеек из M×N пустых матриц;

— cell(M,N,P,…) или cell([M N P …]) создает массив из M×N×P×… пустых матриц.

Пример.

```
cell(2)
ans =
  [ ] [ ]
  [ ] [ ]

C=cell(2,3)
C =
  [ ][ ][ ]
  [ ][ ][ ]
```

Созданные пустые ячейки можно заполнить, используя операции присваивания:

```
C{1,1}=1;
C{1,2}='Привет';
C{2,1}='Hello';
C{2,2}=[1 2; 3 4];
C
C =
 [ 1] 'Привет' [ ]
 'Hello' [2x2 double] [ ]
```

Визуализация массивов ячеек. Для отображения массива ячеек C служит команда celldisp(C). Она дает рекурсивное отображение содержимого массива ячеек C. Например, для ранее созданного массива ячеек A получится следующее:

```
celldisp(A)
A{1,1} =
  Курить вредно!
A{2,1} =
  2.0000 + 3.0000i
A{1,2} =
 1 2
 3 4
A{2,2} =
 Columns 1 through 7
 0 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000
 Columns 8 through 11
 0.7000 0.8000 0.9000 1.0000
```

Для более наглядного графического представления массива ячеек может использоваться команда cellplot:

— cellplot(C) строит структуру массива ячеек C;

— cellplot(C, 'legend') строит структуру массива ячеек C вместе с «легендой» — шкалой стилей представления данных.

Пример (рис. 4).

```
MM{1,1} = 'Prosto text'
MM{1,2} = struct('student', 'Иванов', 'grup', 2, 'estimate', 'good')
MM{2,1} = 4
MM{2,2} = logical([0 2 3; 4 0 6])
cellplot(MM, 'legend')
MM =
 'Prosto text' [1x1 struct ]
 [ 4] [2x3 logical]
```

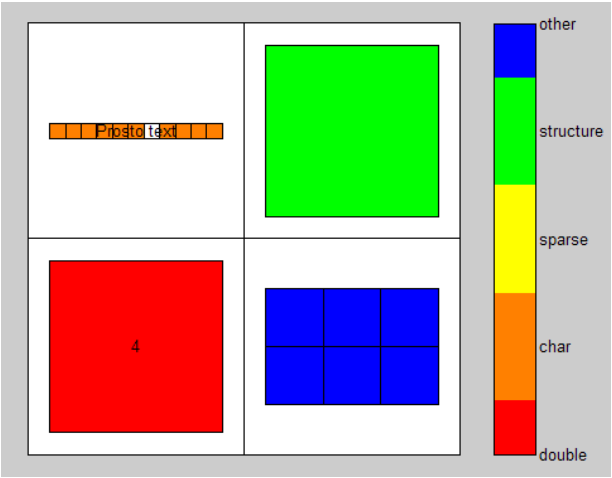

Рис. 4. Пример графического представления массива ячеек

В пакете Matlab имеются функции для преобразования типов данных:

num2cell — преобразовать числовой массив в массив ячеек;

cell2struct — преобразовать массив ячеек в структуру;

struct2cell — преобразовать структуру в массив ячеек.

Пример. Преобразование числового массива в массив ячеек:

```
A = [1 2 3; 4 5 6; 7 8 9]
A1 = num2cell(A)

A =
 1 2 3
 4 5 6
 7 8 9

A1 =
 [1] [2] [3]
 [4] [5] [6]
 [7] [8] [9]
```

Пример. Преобразование массива ячеек в структуру:

```
c = {'tree', 37.4, 'birch'}
f = {'category', 'height', 'name'}
s = cell2struct(c, f, 2)
 c =
 'tree' [37.4000] 'birch'
f =
```

```
 'category' 'height' 'name'
s =
 category: 'tree'
 height: 37.4000
 name: 'birch'
```

Пример. Преобразование структуры в массив ячеек:

```
S = struct('student', 'Иванов', 'grup', 2, 'estimate', 'good')
MM_ja = struct2cell(S)
S =
 student: 'Иванов'
 grup: 2
 estimate: 'good'

MM_ja =
   'Иванов'
   [ 2]
   'good'
```

СПИСОК ЛИТЕРАТУРЫ

1. Галушкин Н.Е. Высокоуровневые методы программирования: язык программирования MatLab: учебник Ч.1. — Ростов н/Д.: Издательство Южного федерального университета, 2011.

2. Колокольникова А.И., Киренберг А.Г. Спецразделы информатики: введение в MatLab: учебное пособие. — М.: Берлин:Директ-Медиа, 2014.

3. Братищев А.В. Руководство к работе с пакетами MATLAB и SIMULINK: учебное пособие. — Ростов н/Д.: ИЦ ДГТУ, 2012.

4. Поршнев С.В. Компьютерное моделирование физических процессов в пакете MATLAB: учебное пособие. — СПб.: Лань, 2011.