الكورس الأول - المحاضرة الثالثة

المرحلة الثالثة - مختبر حاسبات

م.م فاروق خالد شاكر - الفصل الثاني

(المتجهات والمصفوفات)

Chapter 2
Creating Arrays

The array is a fundamental form that MATLAB uses to store and manipulate data. An array is a list of numbers arranged in rows and/or columns. The simplest array (one-dimensional) is a row or a column of numbers. A more complex array (two-dimensional) is a collection of numbers arranged in rows and columns. One use of arrays is to store information and data, as in a table. In science and engineering, one-dimensional arrays frequently represent vectors, and two-dimensional arrays often represent matrices. This chapter shows how to create and address arrays, and Chapter 3 shows how to use arrays in mathematical operations. In addition to arrays made of numbers, arrays in MATLAB can also be a list of characters, which are called strings. Strings are discussed in Section 2.10.

2.1 Creating a One-Dimensional Array (Vector)

A one-dimensional array is a list of numbers arranged in a row or a column. One example is the representation of the position of a point in space in a three-dimensional Cartesian coordinate system. As shown in Figure 2-1, the position of point A is defined by a list of the three numbers 2, 4, and 5, which are the coordinates of the point.

The position of point A can be expressed in terms of a position vector:

$$\mathbf{r}_A = 2\mathbf{i} + 4\mathbf{j} + 5\mathbf{k}$$

where i, j, and k are unit vectors in the direction of the x, y, and z axes, respectively. The numbers 2, 4, and 5 can be used to define a row or a column vector.

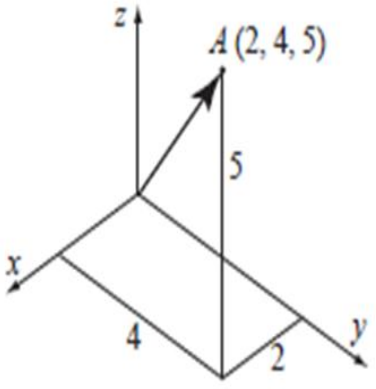

Figure 2-1: Position of a point.

Any list of numbers can be set up as a vector. For example, Table 2-1 contains population growth data that can be used to create two lists of numbers—one of the years and the other of the population values. Each list can be entered as elements in a vector with the numbers

الفصل الثاني

تكوين المصفوفات المصفوفة هي نموذج أساسي تستخدمه MATLAB لتخزين البيانات ومعالجتها .المصفوفة هي قائمة من الأرقام المرتبة في صفوف و / أو أعمدة .أبسط مجموعة أو أحادية البعد) هو صف أو عمود من الأرقام .المصفوفة هي أكثر تعقيدًا (ثنائي الأبعاد (عبارة عن مجموعة من الأرقام المرتبة في صفوف وأعمدة .استخدام واحد المصفوفات تستخدم لتخزين المعلومات والبيانات ، كما في الجدول .في العلوم والهندسة ، المصفوفات أحادية البعد تمثل بشكل متكرر المتجهات والمصفوفات ثنائية الأبعاد غالبًا ما تمثل المصفوفات .يوضح هذا الفصل كيفية إنشاء المصفوفات ومعالجتها ، و يوضح الفصل ٣ كيفية استخدام المصفوفات في العمليات الحسابية .بالإضافة إلى المصفوفات المكونة من الأرقام ، يمكن أن تكون المصفوفات في MATLAB أيضًا قائمة من الأحرف .

variable. This can be done in several ways depending on the source of the information that is used for the elements of the vector. When a vector contains specific numbers that are known (like the coordinates of point *A*), the value of each element is entered directly. Each element can also be a mathematical expression that can include predefined variables, numbers, and functions. Often, the elements of a row vector are a series of numbers with constant spacing. In such cases the vector can be created with MATLAB commands. A vector can also be created as the result of mathematical operations as explained in Chapter 3.

Creating a vector from a known list of numbers:

The vector is created by typing the elements (numbers) inside square brackets [ ].

```
variable_name = [ type vector elements ]
```

Row vector: To create a row vector type the elements with a space or a comma between the elements inside the square brackets.

Column vector: To create a column vector type the left square bracket [and then enter the elements with a semicolon between them, or press the **Enter** key after each element. Type the right square bracket] after the last element.

Tutorial 2-1 shows how the data from Table 2-1 and the coordinates of point *A* are used to create row and column vectors.

Tutorial 2-1: Creating vectors from given data.

```
>> yr=[1984 1986 1988 1990 1992 1994 1996]
```

The list of years is assigned to a row vector named yr.

```
yr =
     1984     1986     1988     1990     1992     1994     1996
>> pop=[127;  130;  136;  145;  158;  178;  211]
```

The population data is assigned to a column vector named pop.

```
pop =
   127
   130
   136
   145
   158
```

```
   178
   211
>> pntAH=[2,   4,   5]
pntAH =
     2     4     5
>> pntAV=[2
4
5]
pntAV =
     2
     4
     5
>>
```

The coordinates of point A are assigned to a row vector called `pntAH`.

The coordinates of point A are assigned to a column vector called `pntAV`. (The **Enter** key is pressed after each element is typed.)

Creating a vector with constant spacing by specifying the first term, the spacing, and the last term:

In a vector with constant spacing the difference between the elements is the same. For example, in the vector $v = 2\ 4\ 6\ 8\ 10$, the spacing between the elements is 2. A vector in which the first term is m, the spacing is q, and the last term is n is created by typing:

`variable_name = [m:q:n]` or `variable_name = m:q:n`

(The brackets are optional.)

Some examples are:

```
>> x=[1:2:13]
x =
     1     3     5     7     9    11    13
>> y=[1.5:0.1:2.1]
y =
   1.5000   1.6000   1.7000   1.8000   1.9000   2.0000   2.1000

>> z=[-3:7]

z =
    -3    -2    -1     0     1     2     3     4     5     6
7
>> xa=[21:-3:6]
```

First element 1, spacing 2, last element 13.

First element 1.5, spacing 0.1, last element 2.1.

First element –3, last term 7. If spacing is omitted, the default is 1.

First element 21, spacing –3, last term 6.

طريقة أنشاء الدالة الخطية (linspaace) .

```
xa =
      21     18     15     12      9      6
>>
```

- If the numbers m, q, and n are such that the value of n cannot be obtained by adding q's to m, then (for positive n) the last element in the vector will be the last number that does not exceed n.
- If only two numbers (the first and the last terms) are typed (the spacing is omitted), then the default for the spacing is 1.

Creating a vector with linear (equal) spacing by specifying the first and last terms, and the number of terms:

A vector with *n* elements that are linearly (equally) spaced in which the first element is *xi* and the last element is *xf* can be created by typing the `linspace` command (MATLAB determines the correct spacing):

```
variable_name = linspace(xi,xf,n)
```

First element — Last element — Number of elements

When the number of elements is omitted, the default is 100. Some examples are:

```
>> va=linspace(0,8,6)        6 elements, first element 0, last element 8.
va =
      0    1.6000    3.2000    4.8000    6.4000    8.0000
>> vb=linspace(30,10,11)     11 elements, first element 30, last element 10.
vb =
    30    28    26    24    22    20    18    16    14    12    10
>> u=linspace(49.5,0.5)      First element 49.5, last element 0.5.
                             When the number of elements is omitted, the default is 100.
u =
  Columns 1 through 10
   49.5000   49.0051   48.5101   48.0152   47.5202   47.0253
46.5303   46.0354   45.5404   45.0455
.............                100 elements are displayed.
Columns 91 through 100
    4.9545    4.4596    3.9646    3.4697    2.9747    2.4798
1.9848    1.4899    0.9949    0.5000
>>
```

تكوين المصفوفات والعمليات التي يمكن أجرائها عليها .

2.2 CREATING A TWO-DIMENSIONAL ARRAY (MATRIX)

A two-dimensional array, also called a matrix, has numbers in rows and columns. Matrices can be used to store information like the arrangement in a table. Matrices play an important role in linear algebra and are used in science and engineering to describe many physical quantities.

In a square matrix the number of rows and the number of columns is equal. For example, the matrix

$$\begin{matrix} 7 & 4 & 9 \\ 3 & 8 & 1 \\ 6 & 5 & 3 \end{matrix} \quad 3 \times 3 \text{ matrix}$$

is square, with three rows and three columns. In general, the number of rows and columns can be different. For example, the matrix:

$$\begin{matrix} 31 & 26 & 14 & 18 & 5 & 30 \\ 3 & 51 & 20 & 11 & 43 & 65 \\ 28 & 6 & 15 & 61 & 34 & 22 \\ 14 & 58 & 6 & 36 & 93 & 7 \end{matrix} \quad 4 \times 6 \text{ matrix}$$

has four rows and six columns. A $m \times n$ matrix has m rows and n columns, and m by n is called the size of the matrix.

A matrix is created by assigning the elements of the matrix to a variable. This is done by typing the elements, row by row, inside square brackets [ ]. First type the left bracket [then type the first row, separating the elements with spaces or commas. To type the next row type a semicolon or press **Enter**. Type the right bracket] at the end of the last row.

```
variable_name=[1st row elements; 2nd row elements; 3rd
               row elements; ... ; last row elements]
```

The elements that are entered can be numbers or mathematical expressions that may include numbers, predefined variables, and functions. *All the rows must have the same number of elements*. If an element is zero, it has to be entered as such. MATLAB displays an error message if an attempt is made to define an incomplete matrix. Examples of matrices defined in different ways are shown in Tutorial 2-2.

Tutorial 2-2: Creating matrices.

```
>> a=[5  35  43;  4  76  81;  21  32  40]
a =
     5    35    43
     4    76    81
    21    32    40
>> b = [7  2  76  33  8
1  98  6  25  6
5  54  68  9  0]
```

A semicolon is typed before a new line is entered.

The **Enter** key is pressed before a new line is entered.

```
b =
     7     2    76    33     8
     1    98     6    25     6
     5    54    68     9     0
>> cd=6; e=3; h=4;
>> Mat=[e, cd*h, cos(pi/3); h^2, sqrt(h*h/cd), 14]
Mat =
    3.0000   24.0000    0.5000
   16.0000    1.6330   14.0000
>>
```

Three variables are defined.

Elements are defined by mathematical expressions.

Rows of a matrix can also be entered as vectors using the notation for creating vectors with constant spacing, or the `linspace` command. For example:

```
>> A=[1:2:11; 0:5:25; linspace(10,60,6); 67 2 43 68 4 13]
A =
     1     3     5     7     9    11
     0     5    10    15    20    25
    10    20    30    40    50    60
    67     2    43    68     4    13
>>
```

In this example the first two rows were entered as vectors using the notation of constant spacing, the third row was entered using the `linspace` command, and in the last row the elements were entered individually.

2.2.1 *The* `zeros`, `ones` *and,* `eye` *Commands*

The `zeros(m,n)`, `ones(m,n)`, and `eye(n)` commands can be used to create matrices that have elements with special values. The `zeros(m,n)` and the `ones(m,n)` commands create a matrix with *m* rows and *n* columns in which all elements are the numbers 0 and 1, respectively. The `eye(n)` command creates a square matrix with *n* rows and *n* columns in which the diagonal elements are equal to 1 and the rest of the elements are 0. This matrix is called the identity matrix. Examples are:

```
>> zr=zeros(3,4)
zr =
     0     0     0     0
     0     0     0     0
     0     0     0     0
>> ne=ones(4,3)
```

```
ne =
     1     1     1
     1     1     1
     1     1     1
     1     1     1
>> idn=eye(5)
idn =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
>>
```

Matrices can also be created as a result of mathematical operations with vectors and matrices. This topic is covered in Chapter 3.

2.3 NOTES ABOUT VARIABLES IN MATLAB

- All variables in MATLAB are arrays. A scalar is an array with one element, a vector is an array with one row or one column of elements, and a matrix is an array with elements in rows and columns.
- The variable (scalar, vector, or matrix) is defined by the input when the variable is assigned. There is no need to define the size of the array (single element for a scalar, a row or a column of elements for a vector, or a two-dimensional array of elements for a matrix) before the elements are assigned.
- Once a variable exists—as a scalar, vector, or matrix—it can be changed to any other size, or type, of variable. For example, a scalar can be changed to a vector or a matrix; a vector can be changed to a scalar, a vector of different length, or a matrix; and a matrix can be changed to have a different size, or be reduced to a vector or a scalar. These changes are made by adding or deleting elements. This subject is covered in Sections 2.7 and 2.8.

2.4 THE TRANSPOSE OPERATOR

The transpose operator, when applied to a vector, switches a row (column) vector to a column (row) vector. When applied to a matrix, it switches the rows (columns) to columns (rows). The transpose operator is applied by typing a single quote ' following the variable to be transposed. Examples are:

```
>> aa=[3   8   1]
aa =
     3     8     1
>> bb=aa'
```

Define a row vector aa.

Define a column vector bb as the transpose of vector aa.

طريقة أجراء عمليات التدوير على عناصر المصفوفة .

```
bb =
     3
     8
     1
>> C=[2 55 14 8; 21 5 32 11; 41 64 9 1]
C =
     2    55    14     8
    21     5    32    11
    41    64     9     1
>> D=C'
D =
     2    21    41
    55     5    64
    14    32     9
     8    11     1
>>
```

Define a matrix C with 3 rows and 4 columns.

Define a matrix D as the transpose of matrix C. (D has 4 rows and 3 columns.)

2.5 ARRAY ADDRESSING

Elements in an array (either vector or matrix) can be addressed individually or in subgroups. This is useful when there is a need to redefine only some of the elements, when specific elements are to be used in calculations, or when a subgroup of the elements is used to define a new variable.

طرق الأستدعاء والاستبدال التي يمكن أجرائها على المتجهات والمصفوفات .

2.5.1 Vector

The address of an element in a vector is its position in the row (or column). For a vector named ve, ve (*k*) refers to the element in position *k*. The first position is 1. For example, if the vector *ve* has nine elements:

ve = 35 46 78 23 5 14 81 3 55

then

ve(4) = 23, *ve*(7) = 81, and *ve*(1) = 35.

A single vector element, *v*(*k*), can be used just as a variable. For example, it is possible to change the value of only one element of a vector by assigning a new value to a specific address. This is done by typing: *v*(*k*) = *value*. A single element can also be used as a variable in a mathematical expression. Examples are:

```
>> VCT=[35 46 78 23 5 14 81 3 55]
VCT =
    35    46    78    23     5    14    81     3    55
>> VCT(4)
```

Define a vector.

Display the fourth element.

```
ans =
    23
>> VCT(6)=273
VCT =
    35    46    78    23     5   273    81     3    55

>> VCT(2)+VCT(8)
ans =
    49
>> VCT(5)^VCT(8)+sqrt(VCT(7))
ans =
   134
>>
```

Assign a new value to the sixth element.

The whole vector is displayed.

Use the vector elements in mathematical expressions.

2.5.2 Matrix

The address of an element in a matrix is its position, defined by the row number and the column number where it is located. For a matrix assigned to a variable *ma*, *ma*(*k*,*p*) refers to the element in row *k* and column *p*.

For example, if the matrix is: $ma = \begin{bmatrix} 3 & 11 & 6 & 5 \\ 4 & 7 & 10 & 2 \\ 13 & 9 & 0 & 8 \end{bmatrix}$

then *ma*(1,1) = 3 and *ma*(2,3) = 10.

As with vectors, it is possible to change the value of just one element of a matrix by assigning a new value to that element. Also, single elements can be used like variables in mathematical expressions and functions. Some examples are:

```
>> MAT=[3 11 6 5; 4 7 10 2; 13 9 0 8]
MAT =
     3    11     6     5
     4     7    10     2
    13     9     0     8
>> MAT(3,1)=20
MAT =
     3    11     6     5
     4     7    10     2
    20     9     0     8
>> MAT(2,4)-MAT(1,2)
ans =
    -9
```

Create a 3 × 4 matrix.

Assign a new value to the (3,1) element.

Use elements in a mathematical expression.