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.

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 placed in a row or in a column.

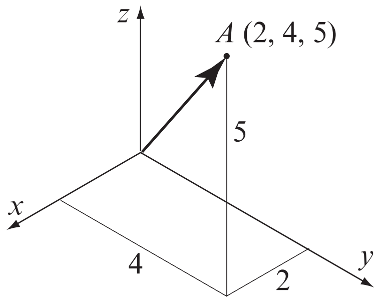

Figure 2-1: Position of a point.

Table 2-1: Population data

Year	1984	1986	1988	1990	1992	1994	1996
Population (millions)	127	130	136	145	158	178	211

In MATLAB, a vector is created by assigning the elements of the vector to a 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
```

Tutorial 2-1: Creating vectors from given data. (Continued)

```
   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.

```
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
u =                               omitted, the default is 100.
  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
>>
```