الكلية	العلوم
القسم	الرياضيات
المادة باللغة الانجليزية	Programming (MATLAB)
المادة باللغة العربية	البرمجة بلغة (ماتلاب)
المرحلة الدراسية	الثانية
اسم التدريسي	صفوت عبدالقادر حمد
عنوان المحاضرة باللغة الانجليزية	Vectors in MATLAB
عنوان المحاضرة باللغة العربية	المتجهات
رقم المحاضرة	الرابعة
المصادر والمراجع	MATLAB A Practical Introduction to Programming and Problem Solving
	MATLAB The Language of Technical Computing
	MATLAB numerical computing

University Of Anbar

College Of Science

Math Department

MATRIX - LABORATORY (MATLAB)

Lecture 4

Vectors in MATLAB

By:

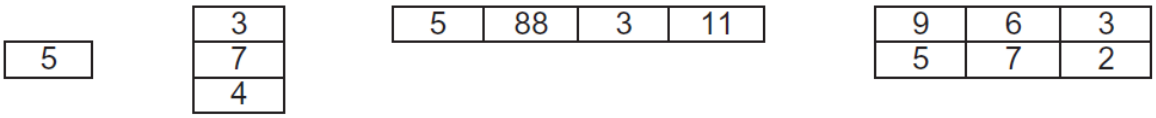
Safwat A. Hamad

For the 2nd stage Math Department

1. Introduction to Vectors

Vectors and matrices are used to store sets of values, all of which are the same type. A matrix can be visualized as a table of values. The dimensions of a matrix are (r * c), where r is the number of rows and c is the number of columns. This is pronounced "r by c." A vector can be either a row vector or a column vector. If a vector has n elements, a row vector would have the dimensions (1 * n) and a column vector would have the dimensions (n * 1).

A scalar (one value) has the dimensions (1*1). Therefore, vectors and scalars are actually just special cases of matrices. Here are some diagrams showing, from left to right, a scalar, a column vector, a row vector, and a matrix:



The scalar is (1 * 1), the column vector is (3 * 1) (three rows by one column), the row vector is (1 * 4) (one row by four columns), and the matrix is (2 * 3) (two rows by three columns). All of the values stored in these matrices are stored in what are called **elements**.

MATLAB is written to work with matrices and so it is very easy to create vector and matrix variables, and there are many operations and functions that can be used on vectors and matrices.

A vector in MATLAB is equivalent to what is called a **one-dimensional array** in other languages. A matrix is equivalent to a **two-dimensional array**. Usually, even in MATLAB, some operations that can be performed on either vectors or matrices are referred to as array operations. The term array is also frequently used to mean generically either a vector or a matrix.

2. Creating Row Vectors

There are several ways to create row vector variables. The most direct way is to put the values that you want in the vector in square brackets, separated by either spaces or commas. For example, both of these assignment statements create the same vector x:

```
>> x = [2 4 8 4]
x =
  2   4   8   4
>> y = [5, 4, 9, 3]
Y =
5   4   9   3
>> disp (x)
x =
  2   4   8   4
>> whos
Name        Size                Bytes  Class     Attributes
  x         1x4                    32  double
  y         1x4                    32  double
```

In addition, we can use one vector in a list for another one. For example

```
>> a = [1 2 3];
>> b = [4 5];
>> c = [a -b];
>> disp (c)
  1   2   3   -4   -5
```

Empty vector

```
>> r = [  ]
```

Note in the Workspace browser that the size of (r) is given as (0 by 0) because (r) is empty. This means r is defined and can be used where an array is appropriate without causing an error; however, it has no size or value.

Making r is empty is not the same as saying (r = 0) (in the latter case (r) has size 1 by 1 or clear r (which removes r from workspace, making it undefined).

The Colon Operator

If, as in the preceding examples, the values in the vector are regularly spaced, the colon operator can be used to iterate through these values. For example, 2:6 results in all of the integers from 2 to 6 inclusive:

```
>> v = 2:6
v =
     2     3     4     5     6
```

Example: 3 to 7

```
>> vr = 3:7
vr =
     3     4     5     6     7
```

In this vector, there are five elements; the vector is a (1 * 5) row vector. **Note** that, in this case, the brackets [ ] are not necessary to define the vector. With the colon operator, a step value can also be specified by using another colon, in the form (first:step:last). For example, to create a vector with all integers from 1 to 9 in steps of 2:

```
>> nv = 1:2:9
nv =
     1     3     5     7     9
```

Another example:

```
>> nr = 10:-2:1
nr =
    10     8     6     4     2
```

Example :

```
>> sn = 0: -2:-6
sn =
     0    -2    -4    -6
```

```
>> sn = 0: -2:-5
sn =
     0    -2    -4
```

According previous example, **note** that when the increment is negative but not equal to -1, the last element is not allowed to be less than the value after the second colon.

```
>> sz = 1:2:6
sz =
     1     3     5
```

According previous example, **note** that when the increment is positive but not equal to 1, the last element is not allowed to exceed the value after the second colon.

In addition, to create empty vector use this example:

```
>> se = 1:0
se =
  1×0 empty double row vector
```

Exercise: How can you use the colon operator to generate the vector shown below?

9 7 5 3 1 ?

The solution is:

```
>> vx = 9: -2: 1
vx =
     9     7     5     3     1
```

Linspace Function

The *linspace* function creates a linearly spaced vector; linspace (**x,y,n**) creates a vector with **n** values in the inclusive range from **x** to **y (number of element)**. If **n** is omitted, the default is 100 points. For example, the following creates a vector with five values linearly spaced between 3 and 15, including the 3 and 15:

```
>> ls = linspace(3,15,5)
ls =
     3     6     9    12    15
```

Vector variables can also be created using existing variables. For example, a new Vector is created here consisting first of all of the values from (*vc1*) followed by all values from (*vc2*):

```
>> vc1 = [4 5 6 3 9];
>> vc2 = [1 6 2];
>> vc3 = [vc1 vc2]
newvc =
     4     5     6     3     9     1     6     2
```

Putting two vectors together like this to create a new one is called concatenating the vectors.

length Function

The length () function in MATLAB returns the length of the largest dimension of an array. For vectors, the length is simply the number of elements. For arrays with more dimensions, the length is the maximum of the sizes of the individual dimensions. For example:

```
>> x = 2     3     4     7
>> length (X)
ans =
      4
```

3. Referring and Modifying Elements

The elements in a vector are numbered sequentially; each element number is called the **index**, or **subscript**. In MATLAB, the indices start at 1. Normally, diagrams of vectors and matrices show the indices. For example, for the variable *newvc* that show below, the indices 1–10 of the elements are shown above the vector:

	1	2	3	4	5	6	7	8	9	10
newvc =	1	3	5	7	9	3	6	9	12	15

A particular element in a vector is accessed using the name of the vector variable and the index or subscript in parentheses. For example, the fifth element in the vector newvc is a 9.

```
>> newvc (5)
ans =
      9
```

The expression newvc (5) would be pronounced “newvc sub 5”, where sub is short for the word subscript. A subset of a vector, which would be a vector itself, can also be obtained using the colon operator. For example, the following statement would get the fourth through sixth elements of the vector newvc and store the result in a vector variable *b*:

```
>> b = newvec(4:6)
b =
  7   9   3
```

Any vector can be used for the indices into another vector, not just one created using the colon operator. The indices do not need to be sequential. For example, the following would get the first, tenth, and fifth elements of the vector newvc:

```
>> newvc([1 10 5])
ans =
  1   15   9
```

The vector [1 10 5] is called an index vector; it specifies the indices in the original vector that are being referenced.

The value stored in a vector element can be changed by specifying the index or subscript. For example, to change the second element from the preceding vector *b* to now store the value 11 instead of 9:

```
>> b(2) = 11
b =
  7   11   3
```

By referring to an index that does not yet exist, a vector can also be extended. For example, the following creates a vector that has three elements. By then assigning a value to the fourth element, the vector is extended to have four elements.

```
>> rv = [3 55 11]
rv =
      3   55   11
>> rv(4) = 2
rv =
      3   55   11   2
```

If there is a **gap** between the end of the vector and the specified element, **0**s are filled in. For example, the following extends the variable *rv* again:

```
>> rv(6) = 13
rv =
      3   55   11   2   0   13
```

Exercises:

1. Create vector called vc1, content element 1 for 10 and increment by 2.
2. How would you create a row vector with elements 1, 3, 5, and 7?
3. Create a row vector *v* with elements ranging from 1 to 20 with a step of 4 ?
4. Create a vector with 10 equally spaced elements between 0 and 1?
5. create a vector v with elements ranging from 12 to 3 with a step of -2 ?
6. create (s_sub) index vector content 4 first element from vector s = 3:3:15 ?
7. replace element (12 to 22) of vector s from (previous point 6) ?
8. found the length of vector s in (previous point 6) ?