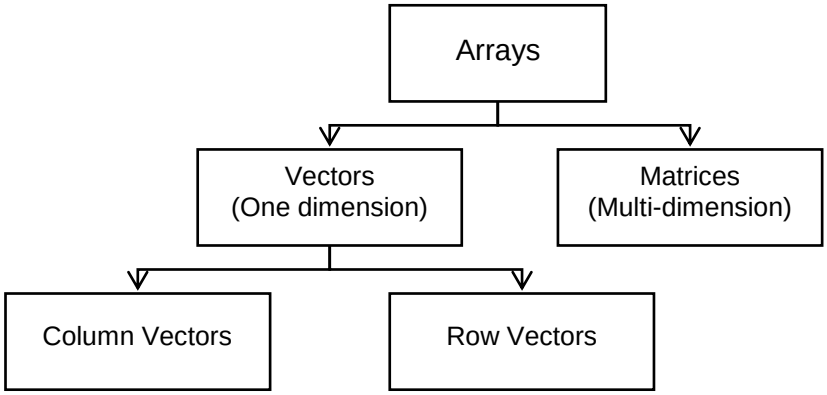
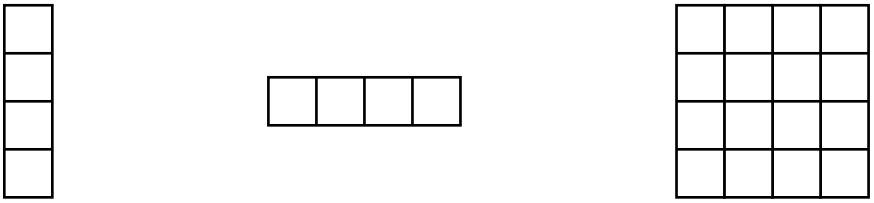
Arrays

An array is a collection of data values organized into rows and columns and known by a single name. Individual elements within an array are accessed by indicating the name of array followed by subscripts in parentheses that identify the row and column of the particular value. Even scalars are treated as arrays; they are simply arrays with only one row and one column. Arrays are the fundamental unit of data in MATLAB.



The ***size*** of an array is the number of rows and columns in a given array.



4x1 column vector 1x4 row vector 4x4 matrix

Creating vectors and matrices in MATLAB

To create a row vector, you simply type the elements inside a pair of square brackets ([ ]) separating them with a space or comma:

```
>> g=[3 4 5] % elements of g were separated by spaces
g =
     3     4     5
>> h=[3,7,9] % elements of h were separated by commas
h =
     3     7     9
```

Use the percent sign (%) to add comments to your statement line. MATLAB will ignore everything after the % symbol.

To create a column vector, you simply type the elements inside a pair of square brackets separating them with a semicolon:

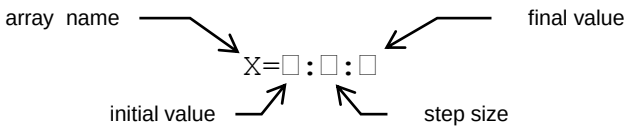
```
>> g=[3;7;9]
g =
     3
     7
     9
```

To create two dimensional array (or matrix), type the matrix elements row by row separating the elements in a given row by spaces or commas and separating the rows by a semicolon:

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

The colon notation (:)

The colon (:) is usually used to generate an array having regularly spaced elements.



- If the step size is unity (i.e., 1), then it is not required to write the step size. Instead, type only the initial value and the final value.
- The step size should be positive if the final value is greater than the initial value.
- The step size should be negative if the final value is less than the initial value.

Examples:

```
>> a=[1:5]
a =
     1     2     3     4     5
>> b=[1:2:7]
b =
     1     3     5     7
>> c=[7:-1:1]
c =
     7     6     5     4     3     2     1
>> d=[0:0.2:1]
d =
         0    0.2000    0.4000    0.6000    0.8000    1.0000
>> e=[40:-10:10]
e =
    40    30    20    10
```

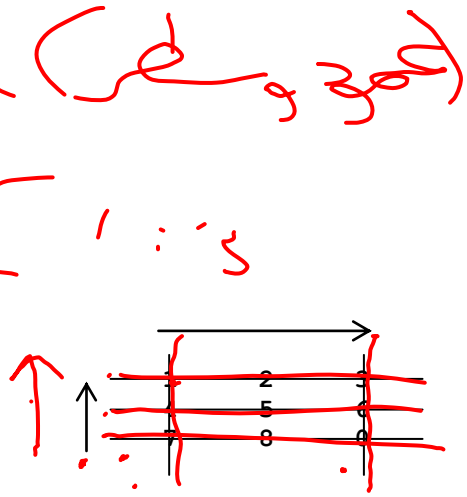
Array addressing

Array indices are the row and column numbers of an element in an array which are used to keep track of the array's elements.

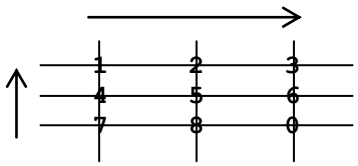
Examples:

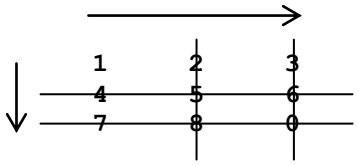
```
>> x=[0:0.1:1]*pi
x =
   0   0.3142   0.6283   0.9425   1.2566   1.5708   1.8850   2.1991   2.5133   2.8274   3.1416
```

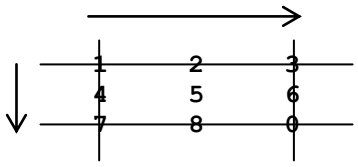
```
>> y=sin(x)
y =
   0    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090    0.5878    0.3090    0.0000
>> x(3) % Show the third element of x
ans =
    0.6283
>> y(5) % Show the fifth element of y
ans =
    0.9511
>> x(1:5) % Show the first to the fifth element of x
ans =
         0    0.3142    0.6283    0.9425    1.2566
>> x(7:end) % Show the 7th through the last element of x
ans =
    1.8850    2.1991    2.5133    2.8274    3.1416
>> x(2:2:7) % Show the 2nd, 4th, and 6th elements of x
ans =
    0.3142    0.9425    1.5708
>> y(3:-1:1) % Show the 3rd, 2nd, and 1st elements of y
ans =
    0.5878    0.3090         0
>> y([8 2 9 1]) % Show the 8th, 2nd, 9th, and 1st elements of y
ans =
    0.8090    0.3090    0.5878         0
>> a=[1:5]
a =
     1     2     3     4     5
>> b=[1:2:9]
b =
     1     3     5     7     9
>> c=[b a] % Create the vector c by combining vectors a and b
c =
     1     3     5     7     9     1     2     3     4     5
>> d=[a(1:2:5) 1 0 1]
d =
     1     3     5     1     0     1
>> a=[1:3;4:6;7:9] % Create matrix a
a =
     1     2     3
     4     5     6
     7     8     9
>> a(3,3)=0 % Set the element a(3,3) to zero
a =
     1     2     3
     4     5     6
     7     8     0
>> b=a(3:-1:1,[1 3])
b =
     7     0
     4     6
     1     3
```

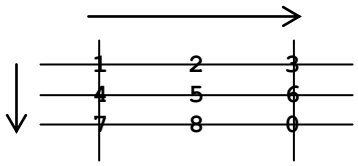


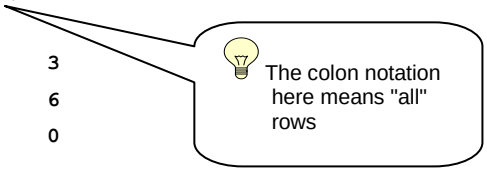
```
>> b=a(3:-1:1,1:3)
b =
     7     8     0
     4     5     6
     1     2     3

>> b=a(2:3,2:3)
b =
     5     6
     8     0
>> c=[1 3] % Create row vector c
c =
     1     3
>> b=a(c,c)
b =
     1     3
     7     0

>> b=a(:,c)
b =
     1     3
     4     6
     7     0
```











Some MATLAB special functions for arrays

MATLAB Function	Description
`zeros(n)`	Generate nxn matrix of zeros
`zeros(n,m)`	Generate nxm matrix of zeros
`ones(n)`	Generate nxn matrix of ones
`ones(n,m)`	Generate nxm matrix of ones
`eye(n)`	Generate nxn identity matrix
`size(A)`	Returns two values specifying the number of rows and columns in array A
`length(A)`	Return the length of a vector or the largest dimension in a given matrix
`linspace(a,b,n)`	Creates a row vector of n regularly spaced values between a and b
`logspace(a,b,n)`	Create a row vector of n logarithmically spaced values between 10^a and 10^b
`max(a)`	Returns the algebraically largest element in A if A is a vector.
	Returns a row vector containing the largest element in each column if A is a matrix
`min(A)`	Returns the algebraically smallest element in A if A is a vector.
	Returns a row vector containing the smallest element in each column if A is a matrix
`sum(A)`	Returns the sum of all elements if A is a vector
	Returns a row vector containing the sum of all elements in each column if A is a matrix

Examples:

```
>> A=[2 4 5;0 1 2; 8 1 3]
A =
     2     4     5
     0     1     2
     8     1     3
>> max(A)
ans =
     8     4     5
>> min(A)
ans =
     0     1     2
>> sum(A)
ans =
    10     6    10
>> size(A)
ans =
     3     3
>> linspace(0,10,7)
ans =
         0    1.6667    3.3333    5.0000    6.6667    8.3333   10.0000
>> logspace(0,1,7)
ans =
    1.0000    1.4678    2.1544    3.1623    4.6416    6.8129   10.0000
```

```
>> B=zeros(3)
B =
     0     0     0
     0     0     0
     0     0     0
>> C=ones(3)
C =
     1     1     1
     1     1     1
     1     1     1

>> D=eye(3)
D =
     1     0     0
     0     1     0
     0     0     1
```

PROBLEMS

1.

a. Use two methods to create the vector x having 100 regularly spaced values starting at 5 and ending at 28.

b. Use two methods to create the vector x having a regular spacing of 0.2 starting at 2 and ending at 14.

c. Use two methods to create the vector x having 50 regularly spaced values starting at -2 and ending at 5.

2.

a. Create the vector x having 50 logarithmically spaced values starting at 10 and ending at 1000.

b. Create the vector x having 20 logarithmically spaced values starting at 10 and ending at 1000.

3. Determine the contents of array a after the following statements are executed:

(a)
```
a=eye(3)
b=[1 2 3]
a(2,:)=b
```

(b)
```
a=ones(2,3)
b=[7 8 9]
a(2,:)=b([3 1 2])
```

4. Given the matrix

$$\mathrm{A} = \begin{bmatrix} 3 & 7 & -4 & 12 \\ -5 & 9 & 10 & 2 \\ 6 & 13 & 8 & 11 \\ 15 & 5 & 4 & 1 \end{bmatrix}$$

a. Find the maximum and minimum values in each column.

b. Evaluate the sum of the second row of A.

c. Find the summation of all elements.

5. Create the following vectors and matrices from the 4x4 matrix A in problem 4:

$$B = \begin{bmatrix} 9 & 10 \\ 13 & 8 \end{bmatrix} \qquad C = \begin{bmatrix} -4 & 12 \\ 10 & 2 \\ 8 & 11 \\ 4 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} 15 & 5 & 4 & 1 \\ 6 & 13 & 8 & 11 \end{bmatrix}$$

$$E = \begin{bmatrix} 5 & 15 \\ 13 & 6 \\ 9 & -5 \end{bmatrix} \qquad F = \begin{bmatrix} 3 & 12 \\ 15 & 1 \end{bmatrix} \qquad G = \begin{bmatrix} 4 \\ 8 \\ 10 \\ -4 \end{bmatrix}$$

$$H = \begin{bmatrix} 2 & 10 & 9 & -5 \end{bmatrix} \qquad I = \begin{bmatrix} 3 & 7 & -4 \\ 6 & 13 & 8 \end{bmatrix} \qquad J = \begin{bmatrix} -5 & 9 & 10 \\ 15 & 5 & 4 \end{bmatrix}$$