



Arrays

Scalars: Variables that represent single numbers, as considered to this point.

Arrays: Variables that represent more than one number. Each number is called an element of the array.

Row and Column Arrays: A row of numbers (called a row vector) or a column of numbers (called a column vector).

Two-Dimensional Arrays: A two-dimensional table of numbers, called a matrix.

Array Indexing or Addressing: Indicates the location of an element in the array.

EX1: consider computing, $y = \sin(x)$ for $0 \leq x \leq \pi$.

x	0	0.1π	0.2π	0.3π	0.4π	0.5π	0.6π	0.7π	0.8π	0.9π	π
y	0	.31	.59	.81	.95	1.0	.95	.81	.59	.31	0

EX2: Generate two vectors, x and y , then display the values such that $x(1)$ and $y(1)$ are on the same line,

```
>> x = 0:4;
```

```
>> y = 5:5:25;
```

```
>> A = [x' y']
```

```
A =
```

```
0 5
```

```
1 10
```

```
2 15
```

```
3 20
```

```
4 25
```

Note that the matrix A has been created from the vectors x and y .



Array Operations

Element-by-Element Array-Array Mathematics

When two arrays have the same dimensions, addition, subtraction, multiplication, and division apply on an element-by-element basis.

Operation	Algebraic Form	MATLAB
Addition	$a + b$	<code>a + b</code>
Subtraction	$a - b$	<code>a - b</code>
Multiplication	$a \times b$	<code>a .* b</code>
Division	$a \div b$	<code>a ./ b</code>
Exponentiation	a^b	<code>a .^ b</code>

Example:

```
>> A = [2 5 6];
```

```
>> B = [2 3 5];
```

```
>> C = A.*B
```

```
C =
```

```
4 15 30
```

```
>> D = A./B
```

```
D =
```

```
1.0000 1.6667 1.2000
```

```
>> E = A.^B
```

```
E =
```

```
4 125 7776
```



Built-in function

A built-in function is part of the MATLAB executable. MATLAB does not implement these functions in the MATLAB language. Although most built-in functions have a .m file associated with them, this file only supplies documentation for the function.

MATLAB includes many built-in functions for math operations. Here are a number of the most important ones.

For these functions, you are specifying input arguments (i.e. 5 is the input argument in the example).

- `sqrt(5)` % square root of 5
- `sin(pi)` % sine of pi radians
- `cos(pi/2)` % cosine of pi/2
- `asin(1)` % arcsine of 1
- `sind(75)` % sine of 75 degrees
- `abs(-5)` % absolute value of -5
- `log(5)` % natural logarithm (base e) of 5
- `log10(5)` % logarithm (base 10) of 5
- `exp(5)` % e^5

If multiple arguments, separate them by commas, as in the example:

- `round(5,3)` % round 5.3 (.5 or greater rounds up)
- `fix(5,3)` % round towards 0
- `floor(5,3)` % round towards -inf



- `ceil(5,3)` % round towards +inf
- `rem(15,2)` % remainder of 15/2
- `mod(15,2)` % similar to rem % but can give negative (congruent) answers
- `sign(x)` % 1 for x>0, 0 for x=0, -1 for x<0

Basic Matrix Functions

MATLAB software provides functions that implement basic statistics. These are:

Function	Description	Function	Description
<code>sum(x)</code>	The sum of the elements of x matrix	<code>Length(x)</code>	Length of largest array dimension
<code>max(x)</code>	Find the largest element in a matrix	<code>Size(x)</code>	returns the sizes of each dimension of array
<code>min(x)</code>	Find the smallest element in a matrix	<code>Sort(x)</code>	Sort array elements
<code>mean(x)</code>	Find the average of the elements of matrix	<code>prod(x)</code>	Product of array elements
<code>median(x)</code>	Find the median value of the elements of matrix	<code>Transpose(x)</code>	Transpose array
<code>diag(x)</code>	Create diagonal matrix or get diagonal elements of matrix	<code>Find(x)</code>	Find indices and values of nonzero elements



Examples

```
>> A=[5 7 9; 4 6 5; 1 9 8]
```

```
>> sum(A) %Create a matrix and compute the sum of the elements in each column.  
ans =
```

```
    10    22    22
```

```
>> sum(A,2) % Create a matrix and compute the sum of the elements in each row.  
ans =
```

```
    21
```

```
    15
```

```
    18
```

```
>> max(A) % find the row vector containing the maximum value of each column.  
ans =
```

```
     5     9     9
```

```
>> max(A,[],2) % compute the largest element in each row.
```

```
ans =
```

```
     9
```

```
     6
```

```
     9
```

```
>> min(A) %find a row vector containing the minimum value of each column.
```

```
ans =
```

```
     1     6     5
```

```
>> min(A,[],2) % compute the smallest element in each row.
```

```
ans =
```

```
     5
```

```
     4
```

```
     1
```



```
>>> mean(A) % compute the mean of each column.
```

```
ans =
```

```
3.3333 7.3333 7.3333
```

```
>>> mean(A,2) % compute the mean of each row.
```

```
ans =
```

```
7
```

```
5
```

```
6
```

```
>>> median(A) % Find the median value of each column.
```

```
ans =
```

```
4 7 8
```

```
>>> median(A,2) % Find the median value of each row.
```

```
ans =
```

```
7
```

```
5
```

```
8
```

```
>>> p=prod(A)
```

```
p =
```

```
20 378 360
```

```
>>> p=prod(A,2)
```

```
p =
```

```
315
```

```
120
```

```
72
```



```
>> v = [2 1 -1 -2 -5];  
>> D = diag(v) % create a matrix with the elements of v on the main diagonal.  
D =  
    2    0    0    0    0  
    0    1    0    0    0  
    0    0   -1    0    0  
    0    0    0   -2    0  
    0    0    0    0   -5  
>> X = diag(A) % Get the elements on the main diagonal of a random 3-by-3 matrix.  
X =  
    5  
    6  
    8  
>> L = length(A)  
L =  
    3  
>> [m,n]=size(A) % return the number of rows (m) and columns (n)  
m = 3  
n = 3  
>> S=sort(A,2) % Create a matrix and sort each of its rows in ascending order.  
S =  
    5    7    9  
    4    5    6  
    1    8    9
```



```
>>> T=transpose(A)    % T has the same elements as A, but the rows of T are the columns  
of A and the columns of T are the rows of A.
```

T =

```
5 4 1  
7 6 9  
9 5 8
```

Find function

If X is a vector, then find returns a vector with the same orientation as X.

If X is a multidimensional array, then find returns a column vector of the linear indices of the result.

```
>>> k = find(A>5)    % Find indices of values greater than 5.
```

$\begin{bmatrix} 5 & 7 & 9 \\ 4 & 6 & 5 \\ 1 & 9 & 8 \end{bmatrix}$	\rightarrow	$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$	\rightarrow	$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$
matrix A		A>5		locations

k =

```
4  
5  
6  
7  
9
```




```
>> [ row , col ] = find(A>5)    % Find indices of values greater than 5.
```

$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$	\rightarrow	$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$		$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$	\rightarrow	$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$
A>5		row location		A>5		col location

```
row =
```

```
1
```

```
2
```

```
3
```

```
1
```

```
3
```

```
col =
```

```
2
```

```
2
```

```
2
```

```
3
```

```
3
```