

Logical Array:

Logical Functions:

MATLAB has a number of useful logical functions that operate on scalars, vectors, and matrices. Examples are given in the following list:-

Function	Description
<code>any(x)</code>	True if any element of a vector is a nonzero number or is logical 1 (TRUE)
<code>all(x)</code>	True if all elements of a vector are nonzero.
<code>find(x)</code>	Find indices of nonzero elements
<code>isnan(x)</code>	True for Not-a-Number
<code>isinf(x)</code>	True for infinite elements.
<code>isempty(x)</code>	True for empty array.

Example:

Let $A=[4\ 9\ 7\ 0\ 5]$,

```
>> any(A)
```

```
ans = 1
```

```
>> all(A)
```

```
ans = 0
```

```
>> find(A)
```

```
ans = 1 2 3 5
```

To remove zero elements from matrix

```
>> B=A(find(A));
```

```
>> B
```

```
B = 4 9 7 5
```

To find the **location** of maximum number of **B**

```
>> find(B==max(B))
```

```
ans = 2
```

Creating a Logical Array:

One way of creating an array of logical is to just enter a true or false value for each element. The [true](#) function returns logical one; the [false](#) function returns logical zero:

- `x = [true, true, false, true, false];`
-

Logical Operations on an Array:

You can also perform some logical operation on an array that yields an array of logical:

- `x = magic(4) >= 9`
- `x =`
- `1 0 0 1`
- `0 1 1 0`
- `1 0 0 1`
- `0 1 1 0`
-

The MATLAB functions that have names beginning with `is` (e.g., [ischar](#), [issparse](#)) also return a logical value or array:

- `a = [2.5 6.7 9.2 inf 4.8];`
-
- `isfinite(a)`
- `ans =`
- `1 1 1 0 1`
-

This table shows some of the MATLAB operations that return a logical true or false.

Function	Operation
true , false	Setting value to true or false
logical	Numeric to logical conversion
& (and), (or), ~ (not), xor , any , all	Logical operations
&&,	Short-circuit AND and OR
== (eq), ~= (ne), < (lt), > (gt), <= (le), >= (ge)	Relational operations
All is* functions, cellfun	Test operations
strcmp , strncmp , strcmpi , strncmpi	String comparisons

Sparse Logical Arrays:

Logical arrays can also be sparse as long as they have no more than two dimensions:

- `x = sparse(magic(20) > 395)`
- `x =`
- (1,1) 1
- (1,4) 1
- (1,5) 1
- (20,18) 1
- (20,19) 1