

Matrices and Arrays

All MATLAB variables are multidimensional *arrays*, no matter what type of data. A *matrix* is a two-dimensional array often used for linear algebra.

Array Creation

To create an array with four elements in a single row, separate the elements with either a comma (,) or a space.

```
a = [1 2 3 4]
a = 1×4
```

1	2	3	4
---	---	---	---

This type of array is a *row vector*.

```
A=[1;2;3]
```

```
A =
```

1
2
3

This type of array is a *column vector*.

To create a matrix that has multiple rows, separate the rows with semicolons.

```
a = [1 3 5; 2 4 6; 7 8 10]
a = 3×3
```

1	3	5
2	4	6
7	8	10

Another way to create a matrix is to use a function, such as ones, zeros, or rand. For example, create a 5-by-1 column vector of zeros.

```
z = zeros(5,1)
z = 5×1
```

0
0
0
0
0

Matrix and Array Operations

MATLAB allows you to process all of the values in a matrix using a single arithmetic operator or function.

```
a + 10
ans = 3×3
```

11	13	15
12	14	16
17	18	20

```
sin(a)
ans = 3×3

    0.8415    0.1411   -0.9589
    0.9093   -0.7568   -0.2794
    0.6570    0.9894   -0.5440
```

To transpose a matrix, use a single quote ('):

```
a'
ans = 3×3

     1     2     7
     3     4     8
     5     6    10
```

You can perform standard matrix multiplication, which computes the inner products between rows and columns, using the `*` operator. For example, confirm that a matrix times its inverse returns the identity matrix:

```
p = a*inv(a)
p = 3×3

    1.0000         0         0
         0    1.0000         0
         0   -0.0000    1.0000
```

Notice that `p` is not a matrix of integer values. MATLAB stores numbers as floating-point values, and arithmetic operations are sensitive to small differences between the actual value and its floating-point representation.

det

Matrix determinant

[collapse all in page](#)

Syntax

`d = det(A)`

Description

`d = det(A)` returns the determinant of square matrix `A`.

Examples

[collapse all](#)

Calculate Determinant of Matrix

Open in MATLAB OnlineCopy Command Copy Code
Create a 3-by-3 square matrix, `A`.

```
A = [1 -2 4; -5 2 0; 1 0 3]
A = 3×3
```

```
     1    -2     4
    -5     2     0
     1     0     3
```

[example](#)

Calculate the determinant of A.

```
d = det(A)
d = -32
```

cross

Cross product

[collapse all in page](#)

Syntax

```
C = cross(A,B)
C = cross(A,B,dim)
```

Description

`C = cross(A,B)` returns the [cross product](#) of A and B.

- If A and B are vectors, then they must have a length of 3.
- If A and B are matrices or multidimensional arrays, then they must have the same size. In this case, the `cross` function treats A and B as collections of three-element vectors. The function calculates the cross product of corresponding vectors along the first array dimension whose size equals 3.

[example](#)

`C = cross(A,B,dim)` evaluates the cross product of arrays A and B along dimension, `dim`. A and B must have the same size, and both `size(A,dim)` and `size(B,dim)` must be 3. The `dim` input is a positive integer scalar.

[example](#)

Examples

[collapse all](#)

Cross Product of Vectors

```
A = [4 -2 1];
B = [1 -1 3];
```

Find the cross product of A and B. The result, C, is a vector that is perpendicular to both A and B.

```
C = cross(A,B)
C = 1x3
```

```
    -5    -11     -2
```

dot

Dot product

[collapse all in page](#)

Syntax

```
C = dot(A,B)
C = dot(A,B,dim)
```

Description

[example](#)

`C = dot(A,B)` returns the [scalar dot product](#) of A and B.

- If A and B are vectors, then they must have the same length.
- If A and B are matrices or multidimensional arrays, then they must have the same size. In this case, the `dot` function treats A and B as collections of vectors. The function calculates the dot product of corresponding vectors along the first array dimension whose size does not equal 1.

[example](#)

`C = dot(A,B,dim)` evaluates the dot product of A and B along dimension, `dim`.

The `dim` input is a positive integer scalar.

Examples

collapse all

Dot Product of Real Vectors

Open in MATLAB OnlineCopy Command Copy Code

Create two simple, three-element vectors.

```
A = [4 -1 2];
```

```
B = [2 -2 -1];
```

Calculate the dot product of A and B.

```
C = dot(A,B)
```

```
C = 8
```

Complex Numbers

Real and imaginary components, phase angles

Functions

abs	Absolute value and complex magnitude
angle	Phase angle
complex	Create complex array
conj	Complex conjugate
cplxpair	Sort complex numbers into complex conjugate pairs
i	Imaginary unit
imag	Imaginary part of complex number
isreal	Determine whether array uses complex storage
j	Imaginary unit
real	Real part of complex number
sign	Sign function (signum function)
unwrap	Shift phase angles

```
z=3-3i;
```

```
w=2+2i;
```

```
z+w
```

```
ans =
```

```
5.0000 - 1.0000i
```

diff

Differentiate symbolic expression or function

[collapse all in page](#)

Syntax

```
Df = diff(f)
```

Description

`Df = diff(f,var)` differentiates `f` with respect to the differentiation parameter `var`. `var` can be a symbolic scalar variable, such as `x`, a symbolic function, such as `f(x)`, or a derivative function, such as `diff(f(t),t)`.

[example](#)
[example](#)

Examples

Differentiate Function

Find the derivative of the function `sin(x^2)`.

```
syms x
sin(x^2);
Df = diff(sin(x^2),x)
Df(x) = 2 x cos(x^2)
```

[example](#)