MORE ON MATLAB BUILT-IN FUNCTIONS

- Decimals in Matlab

- The way decimals are displayed can be formatted using the **format** command

- **format short**

- scaled fixed point format with 5 digits

- integer is followed by four decimal places. example, `3.1416`.

- **format long**

- Scaled fixed point format with 14 to 15 digits after the decimal point for double; and 7 digits after the decimal point for single. For example, `3.141592653589793`.

- **`format short e`**
- Floating point format, with 4 digits after the decimal point. For example, `3.1416e+000`.
- **`format short g`**
- Best of fixed or floating point, with 4 digits after the decimal point. For example, `3.1416`.
-

- **format long e**

- Floating point format, with 14 to 15 digits after the decimal point for double; and 7 digits after the decimal point for single. For example, `3.141592653589793e+000`.

-

- **format long e**

Best of fixed or floating point, with 14 to 15 digits after the decimal point for double; and 7 digits after the decimal point for single. For example, `3.14159265358979`.

- **format rat or rats**

- Ratio of small integers. For example, `355/113`

NB:Format does not affect how Matlab computations are done

Read more on format (>>help format)

Basic Operations

1.ceil

Round toward positive infinity

Syntax

```
B = ceil(A)
```

Description

`B = ceil(A)` rounds the elements of `A` to the nearest integers greater than or equal to `A`.

Examples

```
a = [-1.9, -0.2, 3.4, 5.6,7,
2.4+3.6i]
```

floor

Round toward negative infinity

Syntax

```
B = floor(A)
```

Description

`B = floor(A)` rounds the elements of `A` to the nearest integers less than or equal to `A`. For complex `A`, the imaginary and real parts are rounded independently.

Examples

```
a = [-1.9, -0.2, 3.4, 5.6, 7.0,
2.4+3.6i]
>> floor(a)
```

round

Round to nearest integer

Syntax

```
Y = round(X)
```

Description

`Y = round(X)` rounds the elements of `X` to the nearest integers. For complex `X`, the imaginary and real parts are rounded independently.

Examples

```
a = [-1.9, -0.2, 3.4, 5.6, 7.0,
2.4+3.6i]

>> round(a)
```

fix

Round toward zero

Syntax

```
B = fix(A)
```

Description

`B = fix(A)` rounds the elements of `A` toward zero, resulting in an array of integers. For complex `A`, the imaginary and real parts are rounded independently.

Examples

```
a = [-1.9, -0.2, 3.4, 5.6, 7.0,
2.4+3.6i]
```

>> fix(a)

roundn

Round numbers to specified power of 10

Syntax

```
outnum = roundn(innum)
outnum = roundn(innum,n)
```

Description

`outnum = roundn(innum)` rounds the elements of `innum` to the nearest one-hundredth.

`outnum = roundn(innum,n)` specifies the power of 10 to which the elements of `innum` are rounded. For example, if `n = 2`, round to the nearest hundred (10^2).

NB: if n is negative,it rounds the innum to a specified number of decimal places.

Examples

```
A = 1000*magic(2)/7

A =
        142.8571  428.5714
        571.4286  285.7143

tenths = roundn(fullfig,-1)

tenths =
        142.9000  428.6000
        571.4000  285.7000

units = roundn(fullfig,0)

units =
        143   429
        571   286

tens = roundn(fullfig,1)

tens =
        140   430
        570   290
```

Sqrt- Square root

Syntax

```
B = sqrt(X)

Example

>> s=sqrt(4)
```

imag- Imaginary part of complex number

Syntax

```
Y = imag(Z)
```

real- real part of complex number

Syntax

```
X = real(Z)
```

example

type the following at the command prompt

>> d=sqrt(-1)

>> g=imag(d)

>> h=real(d)

nthroot

Real nth root of real numbers

Syntax

```
y = nthroot(X, n)
```

where n=root type

example

>> nthroot(8,3) →finding cube root of 8.

power

Array power

```
C = power(A,B)

A → the number

B → the power

Example

>> V=power(10,2)

 V=100
```

sum

Sum of array elements

Syntax

```
B = sum(A)
B = sum(A,dim)
```

Description

`B = sum(A)` returns sums along different dimensions of an array.

`B = sum(A,dim)` sums along the dimension of `A` specified by scalar `dim`. The `dim` input is an integer value from 1 to `N`, where `N` is the number of dimensions in `A`. Set `dim` to 1 to compute the sum of each column, 2 to sum rows, etc.

Examples

Try the following at the command prompt

```
>> M = magic(3)

find

1. sum(M)
2. sum(M,1)
3. sum(M,2)
4.sum(diag(M))
5.sum(diag(fliplr(M)))
```

cumsum

Cumulative sum

Syntax

```
B = cumsum(A)
B = cumsum(A,dim)
```

Description

`B = cumsum(A)` returns the cumulative sum along different dimensions of an array.

`B = cumsum(A,dim)` returns the cumulative sum of the elements along the dimension of `A` specified by scalar `dim`. For example, `cumsum(A,1)` works along the first dimension (the columns); `cumsum(A,2)` works along the second dimension (the rows).

Examples

Try the following at the command prompt

```
>> M = magic(3)

find

1. cumsum(M)
2. cumsum(M,1)
3. cumsum(M,2)
4.cumsum(diag(M))
5.cumsum(diag(fliplr(M)))
```

cumprod- Cumulative product

returns the cumulative product along different dimensions of an array.

Syntax

```
B = cumprod(A)
B = cumprod(A,dim)
```

Examples

Try the following at the command prompt

```
>> M = magic(3)

find
```

```
1. cumprod(M)
2. cumprod (M,1)
3. cumprod (M,2)
4.cumprod (diag(M))
5.cumprod (diag(fliplr(M)))
```

max

Largest elements in array

Syntax

```
C = max(A)
C = max(A,B)
C = max(A,[],dim)
[C,I] = max(...)
```

Description

`C = max(A)` returns the largest elements along different dimensions of an array.

`C = max(A,B)` returns an array the same size as `A` and `B` with the largest elements taken from `A` or `B`. The dimensions of `A` and `B` must match, or they may be scalar.

`C = max(A,[],dim)` returns the largest elements along the dimension of `A` specified by scalar `dim`. For example, `max(A,[],1)` produces the maximum values along the first dimension (the rows) of `A`.

min

Smallest elements in array

Syntax

```
C = min(A)
C = min(A,B)
C = min(A,[],dim)
```

Description

`C = min(A)` returns the smallest elements along different dimensions of an array.

`C = min(A,B)` returns an array the same size as `A` and `B` with the smallest elements taken from `A` or `B`. The dimensions of `A` and `B` must match, or they may be scalar.

`C = min(A,[],dim)` returns the smallest elements along the dimension of `A` specified by scalar `dim`. For example, `min(A,[],1)` produces the minimum values along the first dimension (the rows) of `A`.

Examples

Create two 4 - by – 4 matrix ,A and B

Find

a. max(A)

b. min(A)

c.max(B)

d.min(B)

e.max(A,B)

f.min(A,B)

Dimensions of the Matrix

These functions return information about the shape and size of a matrix.

Function

1.`length`

Return the length of the longest dimension. (The length of a matrix or array with any zero dimension is zero.)

Syntax

Q=length(M)

2. `ndims`-Return the number of dimensions.

Syntax

Q=`ndims`(M)

3.`numel`- Return the number of elements.

Syntax

Q=`numel`(M)

Examples

Create two matrices,3-by -4 and 4 – by -3 ,as A and B respectively.

Find

a.length(A) and length(B)

b numel(A) and numel(B)

c ndims(A)

Classes Used in the Matrix

These functions test elements of a matrix for a specific data type.

1. isinteger

Determine if input is an integer array.

Syntax

```
Wb=isinteger(A)
```

2. isnumeric

Determine if input is a numeric array.

Syntax

```
Ar=isnumeric(A)
```

3. isreal

Determine if input is an array of real numbers.

Syntax

```
TF = isreal(A)
```

4. isempty

Determine whether array is empty

Syntax

```
TF = isempty(A)
```

Description

`TF = isempty(A)` returns logical 1 (`true`) if `A` is an empty array and logical 0 (`false`) otherwise.

Examples

```
B = rand(2,2,2);
B(:,:,:) = [];

isempty(B)

ans = 1
```

isnan

Array elements that are NaN

Syntax

```
TF = isnan(A)
```

Description

`TF = isnan(A)` returns an array the same size as `A` containing logical 1 (`true`) where the elements of `A` are `NaN`s and logical 0 (`false`) where they are not.

UNITS CONVERSIONS

1. unitsratio

Unit conversion factors

Syntax

```
ratio = unitsratio(to, from)
```

Description

`ratio = unitsratio(to, from)` returns the number of `to` units per one `from` unit. For example, `unitsratio('cm', 'm')` returns 100 because there are 100 centimeters per meter. `unitsratio` makes it easy to convert from one system of units to another. Specifically, if `x` is in units `from` and

```
y = unitsratio(to, from) * x
```

example

convert 30meters to feet

>> e=unitsratio(‘ft’,’m’)*30

Units of Angle

`unitsratio` recognizes the following identifiers for converting units of angle:

Unit Name	String(s)
radian	`'rad', 'radian(s)'`
degree	`'deg', 'degree(s)'`

Syntax

```
Ratio = unitsratio(to, from)
```

Example

Convert 60degrees to radian

>> e=unitsratio(‘rad’,’deg’)*60

You can also use

rad2deg

Convert angle units from radians to degrees

Syntax

```
anglout = rad2deg(anglin)
```

Description

`anglout = rad2deg(anglin)` converts angles input in radians to the equivalent measure in degrees.

Example

Convert 60 degrees to radians using rad2deg

>> d=rad2deg(60)

deg2rad

Convert angles from degrees to radians

Syntax

```
angleOut = deg2rad(angleIn)
```

Description

`angleOut = deg2rad(angleIn)` converts angles input in degrees to the equivalent measure in radians.

Convert 1.0471975511966 radian to

degrees using rad2deg

e=rad2deg(1.0471975511966

)

CONVERTING DEG,MIN SEC TO DECIMAL DEGREES

1.dms2degrees

Convert degrees-minutes-seconds to degrees

Syntax

```
angleInDegrees = dms2degrees(DMS)
```

Description

`angleInDegrees = dms2degrees(DMS)` converts angles from degree-minutes-seconds representation to values in degrees which may include a fractional part (sometimes called "decimal degrees").

degrees2dms

Convert degrees to degrees-minutes-seconds

Syntax

```
DMS = degrees2dms(angleInDegrees)
```

Description

`DMS = degrees2dms(angleInDegrees)` converts angles from values in degrees which may include a fractional part (sometimes called "decimal degrees") to degree-minutes-seconds representation. The input should be a real-valued column vector.

degrees2dm

Convert degrees to degrees-minutes

Syntax

```
DM = degrees2dm(angleInDegrees)
```

rad2dms, rad2dm

Convert angles from radians to deg:min or deg:min:sec encoding

Syntax

```
anglout = rad2dms(anglin)
angleout = rad2dm(anglin)
```

Modulo Arithmetic

mod

Modulus after division

Syntax

```
M = mod(X,Y)
```

`Where:`

`X → the number`

`Y → the mod`

`NB:` The inputs `X` and `Y` must be real arrays of the same size, or real scalars.

Examples

```
mod(13,5)
ans =
     3

mod([1:5],3)
ans =
     1     2     0     1     2

mod(magic(3),3)
ans =
     2     1     0
     0     2     1
     1     0     2
```

REMENDER AFTER DIVISION

rem

Remainder after division

Syntax

```
R = rem(X,Y)
```

Where:

X → the number to be divided (dividend)

Y → the divisor.

NB: The inputs `X` and `Y` must be real arrays of the same size, or real scalars.

The following are true by convention:

- `rem(X,0)` is `NaN`
- `rem(X,X)` for `X~=0` is `0`
- `rem(X,Y)` for `X~=Y` and `Y~=0` has the same sign as `X`.

examples

rem(3,0),rem(3,3),rem(7,5)