format

Set display format for output

Graphical Interface

As an alternative to `format`, use preferences. Select **Preferences** from the **File** menu in the MATLAB desktop and use **Command Window** preferences.

Syntax

```
format
format type
format('type')
```

Description

Use the `format` function to control the output format of numeric values displayed in the Command Window.

> **Note** The `format` function affects only how numbers are displayed, not how MATLAB computes or saves them.

`format` by itself, changes the output format to the default appropriate for the class of the variable currently being used. For floating−point variables, for example, the default is `format short` (i.e., 5−digit scaled, fixed−point values).

`format type` changes the format to the specified *`type`*. The tables shown below list the allowable values for *`type`*.

`format('type')` is the function form of the syntax.

The tables below show the allowable values for *`type`*, and provides an example for each type using `pi`.

Use these format types to switch between different output display formats for floating−point variables.

Type	Result	Example
`short`	Scaled fixed point format, with 5 digits	`3.1416`
`long`	Scaled fixed point format, with 15 digits for double; 7 digits for single.	`3.14159265358979`
`short e`	Floating point format, with 5 digits.	`3.1416e+000`
`long e`	Floating point format, with 15 digits for double; 7 digits for single.	`3.141592653589793e+000`
`short g`	Best of fixed or floating point, with 5 digits.	`3.1416`

`long g`	Best of fixed or floating point, with 15 digits for double; 7 digits for single.	`3.14159265358979`
`short eng`	Engineering format that has at least 5 digits and a power that is a multiple of three	`3.1416e+000`
`long eng`	Engineering format that has exactly 16 significant digits and a power that is a multiple of three	`3.14159265358979e+000`

Use these format types to switch between different output display formats for all numeric variables.

Value for type	Result	Example
`+`	+, −, blank	`+`
`bank`	Fixed dollars and cents	`3.14`
`hex`	Hexadecimal (hexadecimal representation of a binary double−precision number)	`400921fb54442d18`
`rat`	Ratio of small integers	`355/113`

Use these format types to used to affect the spacing in the display of all variables.

Value for type	Result	Example
`compact`	Suppresses excess line feeds to show more output in a single screen. Contrast with `loose`.	`theta = pi/2 theta= 1.5708`

`loose`	Adds linefeeds to make output more readable. Contrast with `compact`.	`theta = pi/2`
		
		`theta=`
		
		`1.5708`

Remarks

Computations on floating−point variables, namely `single` or `double`, are done in appropriate floating−point precision, no matter how those variables are displayed. Computations on integer variables are done natively in integer.

MATLAB always displays integer variables to the appropriate number of digits for the class. For example, MATLAB uses three digits to display numbers of type `int8` (i.e., −128:127). Setting`format` to `short` or `long` does not affect the display of integer variables.

The specified format applies only to the current MATLAB session. To maintain a format across sessions, use MATLAB preferences.

To see which `type` is currently in use, type

```
get(0,'Format')
```

To see if `compact` or `loose` formatting is currently selected, type

```
get(0,'FormatSpacing').
```

Examples

Example 1

Change the format to `long` by typing

```
format long
```

View the result for the value of `pi` by typing

```
pi
ans =
   3.14159265358979
```

View the current format by typing

```
get(0,'format')
ans =
   long
```

Set the format to `short e` by typing

```
format short e
```

or use the function form of the syntax

```
format('short','e')
```

Example 2

When the format is set to `short`, both `pi` and `single(pi)` display as 5−digit values:

```
format short

pi
ans =
    3.1416

single(pi)
ans =
    3.1416
```

Now set format to `long`, and `pi` displays a 15−digit value while `single(pi)` display an 8−digit value:

```
format long
```

```
pi
ans =
    3.14159265358979

single(pi)
ans =
    3.1415927
```

Example 3

Set the format to its default, and display the maximum values for integers and real numbers in MATLAB:

```
format

intmax('uint64')
ans =
   18446744073709551615

realmax
ans =
  1.7977e+308
```

Now change the format to hexadecimal, and display these same values:

```
format hex

intmax('uint64')
ans =
   ffffffffffffffff

realmax
ans =
   7fefffffffffffff
```

The hexadecimal display corresponds to the internal representation of the value. It is not the same as the hexadecimal notation in the C programming language.

Example 4

This example illustrates the `short eng` and `long eng` formats. The value assigned to variable `A` increases by a multiple of 10 each time through the `for` loop.

```
A = 5.123456789;

for k=1:10
   disp(A)
   A = A * 10;
end
```

The values displayed for `A` are shown here. The power of 10 is always a multiple of 3. The value itself is expressed in 5 or more digits for the `short eng` format, and in exactly 15 digits for `long eng`:

```
format short eng                format long eng

   5.1235e+000                   5.12345678900000e+000
  51.2346e+000                  51.2345678900000e+000
 512.3457e+000                 512.345678900000e+000
   5.1235e+003                   5.12345678900000e+003
  51.2346e+003                  51.2345678900000e+003
 512.3457e+003                 512.345678900000e+003
   5.1235e+006                   5.12345678900000e+006
  51.2346e+006                  51.2345678900000e+006
 512.3457e+006                 512.345678900000e+006
   5.1235e+009                   5.12345678900000e+009
```

Algorithms

If the largest element of a matrix is larger than 10^3 or smaller than 10^{-3}, MATLAB applies a common scale factor for the short and long formats. The function `format +` displays +, −, and blank characters for positive, negative, and zero elements. `format hex` displays the hexadecimal representation of a binary double−precision number. `format rat` uses a continued fraction algorithm to approximate floating−point values by ratios of small integers. See `rat.m` for the complete code.

See Also

disp, display, isnumeric, isfloat, isinteger, floor, sprintf, fprintf, num2str, rat, spy

for fplot