Texts

Jiseok Chae

Department of Mathematical Sciences
KAIST

Week 13

Contents

The main usage of MATLAB is to use it as a mathematical software, but still, there are cases where you need to manipulate texts.

For example, a data you want to read in can contain qualitative information written in text instead of numbers, or you may want to enhance your plots by including some written information .

Technically, there are two types of text data in MATLAB: *character vectors* and *strings*.

```
>> s1 = 'Hello?'; % Using ' ' :  A character vector
>> s2 = "No thanks..."; % Using " " : A string
>> class(s1)

ans =

    'char'

>> class(s2)


ans =

    'string'
```

Technically, there are two types of text data in MATLAB: *character vectors* and *strings*.

The difference is that a character vector is a "vector" (that is, an array) of characters, while a string is a single item by itself.

A character is (roughly) a glyph in a computer, including letters, digits, and punctuation marks. Whitespaces are also characters.

However, let us not bother distinguishing between them. We will only focus on character vectors, but we will still just call them strings.

Recall that a string created by single quotation marks is a character *vector*. Several vector operations are directly overloaded.

```
>> s1 = 'Algebra';
>> numel(s1) % = length(s1)

ans =

     7

>> s0 = 'Linear';
>> s = [s0 ' ' s1]

s =

    'Linear Algebra'
```

To really convince you that character vectors are really vectors, we present you the following example:

```
>> s1'

ans =

  7×1 char array

    'A'
    'l'
    'g'
    'e'
    'b'
    'r'
    'a'
```

To make an array of strings, you can use the function `char`.

```
>> s2 = 'Analysis'; s3 = 'Topology'; s4 = 'Geometry';
>> courses = char(s1, s2, s3, s4)

courses =

  4×8 char array

    'Algebra '
    'Analysis'
    'Topology'
    'Geometry'
```

However, note that `s1` is now padded with whitespace to match the length of the other strings.

The function `deblank` removes all the trailing blanks, and `strtrim` removes all the leading and trailing blanks.

```
>> s5 = '    Statistics   ';
>> deblank(s5)

ans =

    '    Statistics'

>> strtrim(s5)

ans =

    'Statistics'
```

Using the equality operator == will compare strings character by character, and only work when two strings have the same length.
The more "natural" comparison is done by `strcmp`.

```
>> t1 = 'Numerical'; t2 = 'Analysis';
>> s2 == t2

ans =

  1×8 logical array

   1   1   1   1   1   1   1   1

>> strcmp(s2, t2)

ans =

  logical

   1
```

Converting between numbers and strings are done by `num2str` and `str2num`.

```
>> disp(pi)
    3.1416
>> num2str(pi)

ans =

    '3.1416'

>> minusone = str2num('-1')

minusone =

    -1
```

Contents

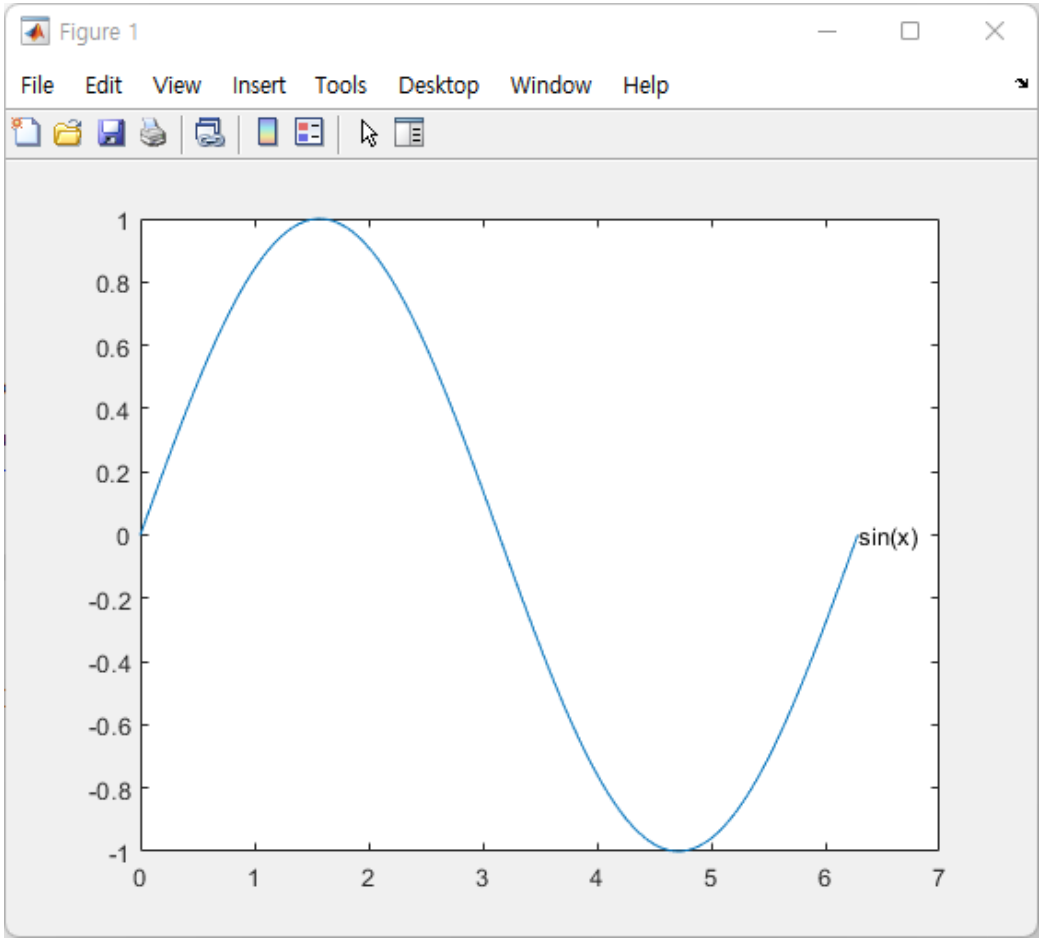
The function `text` *adds* a string onto a plot. (It does not create a plot on its own.)

Let us try executing the following script.

plot_with_name.m

```
x = linspace(0, 2*pi, 150);
y = sin(x);
plot(x, y);
text(2*pi, 0, 'sin(x)');
```

With the plot of the graph of $y = \sin x$, the text sin(x) appears on the right of the point $(2\pi, 0)$ as specified.

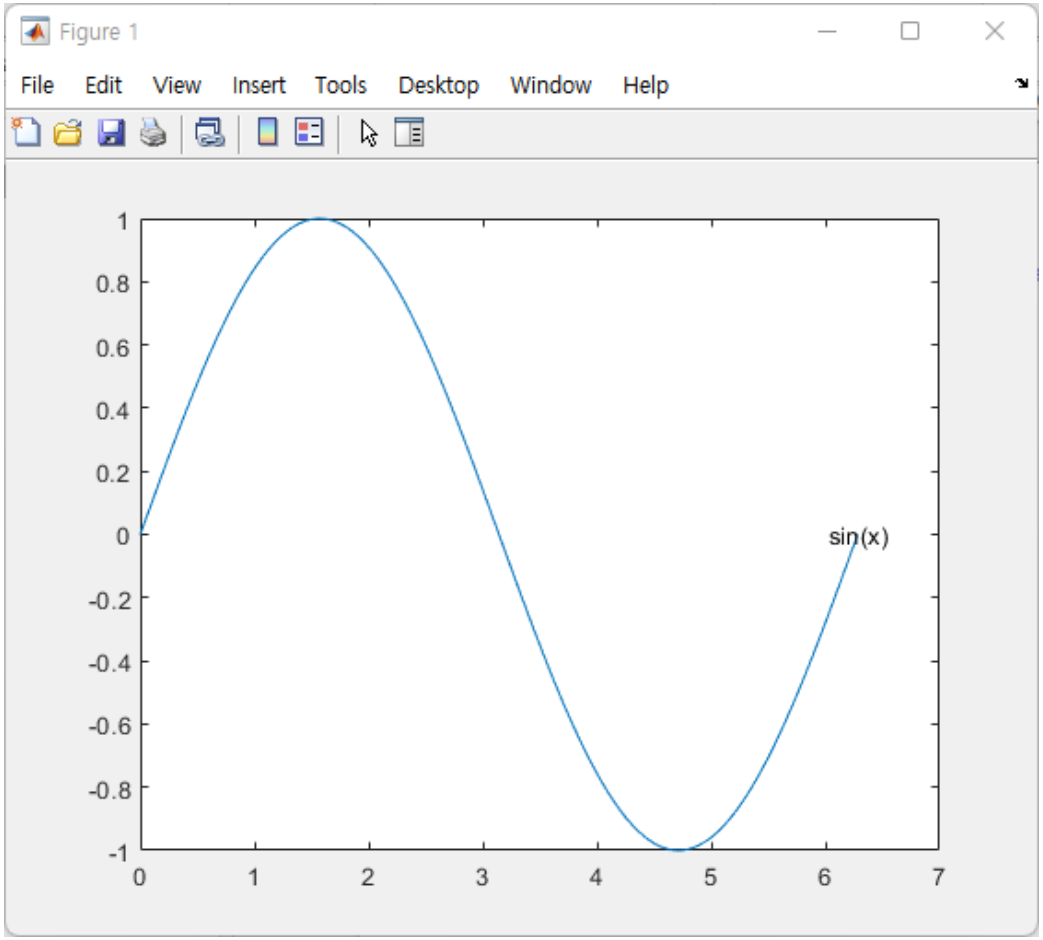


To make the text to be centered at the specified point, you should add such options. Let us change the last line of the script as follows:

plot_with_centered_name.m

```
x = linspace(0, 2*pi, 150);
y = sin(x);
plot(x, y);
text(2*pi,0,'sin(x)','HorizontalAlignment','center');
```

Now the text sin(x) appears centered at the point $(2\pi, 0)$.



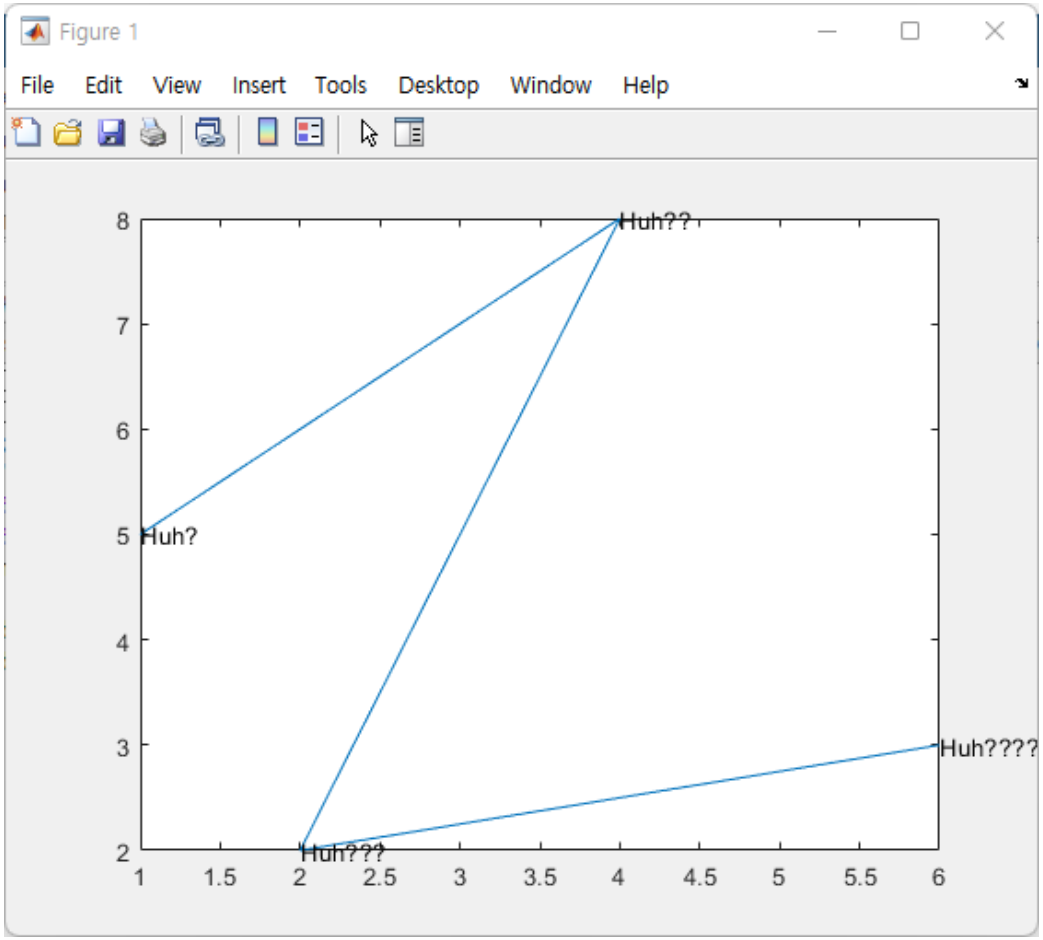
Multiple texts can be added simultaneously.

what_is_happening.m

```
x_coords = [1, 4, 2, 6];
y_coords = [5, 8, 2, 3];
s1 = 'Huh?';
s2 = 'Huh??';
s3 = 'Huh???';
s4 = 'Huh????';
str_arr = char(s1, s2, s3, s4);

plot(x_coords, y_coords);
text(x_coords, y_coords, str_arr);
```

Also note that some texts are outside of the axes, as they are added after drawing the plot.



Thank you!