Lab 05: Formatting Output and Generating Tables

Math 3341: Introduction to Scientific Computing Lab Spring 2018

When Matlab performs calculations, it uses a default level of precision (we'll discuss this in a later lab). This is different than how Matlab actually displays values. We can either specify how all numerical information is displayed in the Command Window, or we can specify formatting for particular values. Options discussed here only affect how it appears when printed.

Format of Output in Command Window

The format of numerical output can be changed by entering `format <formatType>` directly into the command window. Table 1 lists some of the common format types. Additional options can be found in the Matlab documentation.

You may have already noticed the fact many programming languages use `e` to represent 10 in scientific notation. If you want to use e^x you must instead use `exp(x)`. For example, `2e+06` $= 2 \times 10^6$.

Table 1

Command	Description	Example
`format <type>`	set format type	`format long`
`format`	resets command window to default	
`short`	(default) decimal format; 4 digits after the decimal point	`3.1416`
`long`	decimal format; 15 digits after the decimal point	`3.141592653589793`

Keep in mind that changing the format in the command window will change how all values are output to the command window, unless otherwise specified in your code. In many cases, this will NOT be the behavior that you want.

Printing Text Along with Numerical Output

We can specify how MATLAB prints specific values to the command window using the `fprintf` function. You have already seen how text can be displayed using the `disp` function. This is fine if we want to print a single line of text. If we want to include numerical information we will need to use `fprintf`. The two commands are explained in Table 2.

Table 2

Command	Description
`fprintf('format', listOfVars)`	format is a string that specifies precise output for each variable or expression `listOfVars`. Must use `\n` to call a newline.
`disp(<varName>)`	Displays the variable `varName`. Can also output a basic string of characters. Does not need a newline.

Specifying Precision

The *precision* of a value is equivalent to how many decimal places you want to display. Using `fprintf` we can specify the output precision of a particular value. For example,

```
'%8.2f'
```

tells Matlab to set 8 characters worth of space and display the value as fixed point decimal with 2 decimal places. Options for format types is given in Table 3.

Table 3: `fprintf` format types

Format	Description
`'%d'`	prints a decimal or integer value. Ideal for integer output
`'%f'`	prints a fixed point value
`'%e'`	prints value with scientific notation
`'%g'`	prints whichever is shorter, `'%e'` or `'%f'`
`'%s'`	prints formatted string (i.e. text), use in list or table of values.
`'%+f'`	prints a fixed point value with its sign
`'%-12f'`	left-justified text
`'\n'`	starts next output on a new line

Specifying Field Width

The number in front of the letter and decimal point indicates the *field width*. This is the exact number of characters worth of space you want for your value (regardless of how much space it actually occupies).

Use the field width to adjust columns so that entries line up properly.

Text Alignment

By default Matlab right justifies all output within the field width. To get left justified text use a `-` in front of the value. For example

```
b = 0.003;
fprintf('right justified: %10.3f\n', b)
fprintf(' left justified: %-10.3f\n', b)
```

```
right justified:      0.003|
 left justified: 0.003     |
```

Special Symbols

Just as with LaTeX, certain text symbols like `%, \, '` are also part of Matlab's syntax. In order to get these symbols to print in `fprintf` we need to specify that we want the actual symbol printed.

Table 4

Command	Description
`''`	prints a single quotation mark'
`\\`	prints a backslash \
`%%`	prints percent sign %

Using `fprintf` instead of `disp`

You may have noticed that Matlab places red squiggly lines under = if you do not suppress output. Why is this? Often, you want values and information to print in a specific way and at a specific time. Having variables just print to the screen may be simple, but it results in messy output or hard to follow code. You should now start using `fprintf` instead of `disp` or just letting variables be printed to the command window as is. What exactly does the difference between using `disp` and `fprintf` actually look like?

Example Using `disp`

In the first homework assignment you needed to display several lines of output. Most likely, you used `disp` to display text, and then had MATLAB just print the variable. This output looked something like what is shown in Figure 2.

The code that generates this is shown below

Figure 1

```
disp('Convergence was reached:')
disp('Root is')
xM
disp('Function value at root is')
f(xM)
disp('Number of iterations')
j
```

Figure 2

```
Convergence was reached:
Root is

xM =

    1.8779

Function value at root is

fx =

   2.3824e-06

Number of iterations

j =

    19
```

Example Using `fprintf`

Using `fprintf` we can make our output look 'nicer' as shown in Figure 3. The lines of code that produced this output are shown in Figure 4.

Figure 3

```
Convergence was reached:
Root is 1.877942
f(1.877942) = 0.000002
Number of iterations: 19
```

Figure 4

```
disp('Convergence was reached:')
fprintf('Root is %f \n',xM)
fprintf('f(%f) = %f \n',xM,f(xM))
fprintf('Number of iterations: %d \n',j)
```

Recall that for the methods you have coded so far, you have always included a value for *tolerance* i.e. the value $10^(- 5)$ in most cases. When formatting output for your function value, you could set output to only display precision up to this same value. Anything less than this value will print as zero (even though we know the value may not be exactly zero).

Additional Functions and Commands

This lab also makes use of MATLAB's random number generators. These functions are listed in Table 5.

Table 5

Command	Description
`rand(n)`	outputs an $n \times n$ array of random values
`rand(m,n)`	outputs an array of random values in interval $(0, 1)$ with m rows and n columns.
`(b-a)*rand(m,n)+ a`	outputs an m by n array of uniformly distributed values on the interval (a, b)

Note: "uniformly distributed" means that any one of the entries can take any value in the interval with the same probability.

Lab Exercises

Download the script file named `lab05script.m`. Run the script file and note how the output is formatted in the command window.

I. Formatting Numerical Values

1. In Part I of the script file, using the first `fprintf` statement as a guide print the value of π with the following format requirements.
   (a) Using `f`, with 5 decimal places.
   (b) Using `e`, with 3 decimal places.
   (c) Using `e`, with 7 decimal places.
   (d) Using `g`, with no decimal places specified.
2. Now print the value of $\pi/1000$ where the output meets the following formatting requirements.
   (a) Using `e`. (b) Using `f`. (c) Using `g`.
3. Print the value 2 where the output meets the following formatting requirements.
   (a) Using `d`. (b) Using `e`. (c) Using `f`. (d) Using `g`.

II. Choosing the Right Format

1. Uncomment the 4 lines of code in Part II of the script file and run the script file. We can see that our choice of format for the output matters. Depending on what we are calculating, we will need to choose appropriate amount of decimal places to be displayed or we can use valuable information.
2. Duplicate the last line of code. Change each of the output formats to use `g` instead. What do you notice?

III. Formatting Output with Strings

1. Below is a table of information. The first line of this table was printed using the following ML syntax: `fprintf('%14s %8.1fn','Temperature',301.3)`

   This first line is given in your script file. Use it as a guide to add the next two lines of the table. Your output should look exactly the same as in Figure 5.

Figure 5

```
   Temperature:      301.3
      Energy(J):   1.24e+06
Density(kg/m3):       1.21
```

2. Copy/paste this table, but change the alignment so entries are left-justified.

IV. Formatting Tables

1. Uncomment the code in `Part IV`. You will see some lines of code which generate 3 vectors with random values as entries. The `fprintf` statements and `for loop` generate a table of these entries. Run the script file to see what the output looks like.

Figure 6

iter	x	y	z
1	192.39	4.43237e+01	−16.66161
2	−149.19	1.62984e+02	76.37731
3	−107.10	3.53631e+02	282.47846
4	−190.55	2.19253e+02	−50.38963

2. Without changing the number of decimal places displayed, adjust the field width headings and/or entries so that the columns line up properly. In other words, the table should look something like the one in Figure 6 (note actual entries may vary).

After you have finalized your results, call `diary('lab05output.txt')`, run your script file once more with the complete output, then call `diary off`.