Lab 1. Using MATLAB as a Calculator

April 10, 2020

0.0.1 TO DO: Rewrite code/outputs MATLAB -> Python/Numpy/Jupyter

The College of Staten Island, Department of Mathematics, MTH229

Lab 1. Using MATLAB as a Calculator

1 Introduction

This project is designed to give you a brief introduction to the MATLAB software which will be used to help carry out your computer projects during this semester. This software is especially designed for mathematical, scientific and engineering applications.

Many of you probably have already used a scientific or graphics calculator like the TI-84. In its simplest application, MATLAB can be used just like a graphing calculator. In this project we learn how to turn "math into MATLAB." That is, we learn how to ask MATLAB math questions such as "what is 2+2?"

1.1 New Python topics

- Arithmetic Operations and Precedence Rules
- Function Evaluation
- Command Line Editor (correcting errors)
- Arrays and Tables
- Printing Text
- Resizing and Moving Windows

1.2 New Python commands

- `;` – The semicolon suppresses the output.
- `x=a:h:b` – This construct creates sequences of numbers.
- `[x;y]'` – This construct creates a nice tabular look to lists of numbers.

1.3 On-line resources for MATLAB and the MATLAB projects

This project, and others, are available in an on-line format. Go to the following url for more information:

http://www.math.csi.cuny.edu/matlab

Figure 1: Initial MATLAB window contains Command Window, Workspace viewer, and Command History panes.

MATLAB is started by double clicking its desktop icon (on a typical installation of MATLAB on Windows, other installations may vary). Figure 1 shows a typical startup window. It consists of several panes. Additionally other windows, such as a help window or plot window, may appear.

The visible panes in the figure are the **Command Window** which appears on the right side of the window. This window contains the command line where we type commands for MATLAB to interpret. This will be our main method to interact with MATLAB. Additionally, in the upper left pane is a Workspace viewer which shows the variables that are present in the MATLAB session, and in the lower left pane the Command History window displays the commands you have entered into the command window.

Commands are typed after the **prompt**: `>>`. After they are typed in, the **Enter** key is hit to send the command to the MATLAB interpreter. MATLAB either shows the answer or will respond with an error message indicating why it couldn't do what was requested. Multiple commands may be entered on the same line if the commands are separated by a semicolon `;`, or a comma `,`. As well, a semicolon at the end of a command causes its output not to be displayed.

2 Arithmetic Operations

We begin our explorations with MATLAB by learning how to do simple arithmetic operations:

For instance, adding 2 + 2 is done with

```
>> 2 + 2
ans = 4
```

After typing in the command `(2 + 2)`, we typed the Enter key. MATLAB responded with an answer of 4.

Some other examples are shown below:

	symbol	example	MATLAB response
addition	`+`	`>> 3+4`	`ans = 7`
subtraction	`-`	`>> 5-9`	`ans = -4`
multiplication	`*`	`>> 5*9`	`ans = 45`
division	`/`	`>> 8/9`	`ans = 0.8889`
exponentiation	`**`	`>> 5**3`	`ans = 125`

The only symbol that may need learning is the `**` for powers.

2.1 Assignment

When the command `2 + 2` is typed above, MATLAB evaluates it and responds with an answer of 4. It then forgets about this calculation because we didn't ask for it to be saved. In order to save values we can assign the output to a variable. After doing this, we can refer to the output by the variable name.

In the example below, we assign the value 18 to the variable a, the value 21 to the variable b, and the value $18 - 21 = -3 = a - b$ to the variable c.

[1]:
```
a = 18
b = 21
c = a - b
```

Once a value is assigned to a variable, MATLAB remembers it until that variable is reassigned.

Try predicting the responses to the following MATLAB commands to check your understanding.

[2]:
```
a = 3; b = 4;c = 5;
a=b*c
b/a
a=a-18
a**b
```

[2]:
```
16
```

When you make an assignment, the variable name appears in the Workspace viewer which appears in the upper left pane of the initial MATLAB window. You can view the contents of the variable by double clicking it in the workspace viewer. Otherwise, you can enter just the variable name in the command line and its contents will be displayed.

2.2 Exercise 1

What is the output of the following commands:

[3]:
```
a=3
b=4
c=5
a + b/c
```

[3]:
```
3.8
```

(Q1) Circle one: 1. 3 4/5 1. 1.6 1. 3.8 1. 7/5

Example 1: Assigning variables can simplify matters when used wisely. For example, when evaluating

$$\frac{(2-3)-(-3)}{(-1)+2}$$

One way is to write out the whole expression at once:

[4]:
```
((2 - 3) - (-3))/((-1) + 2)
```

[4]:
```
2.0
```

There are many parentheses that are needed to get this right. This can make finding errors in our work tough. If we use spaces (which are ignored) and intermediate names, we can reduce the chance of a typing error:

[5]:
```
top = (2 - 3) - (-3)
bottom = (-1) + 2
top/bottom
```

[5]:
```
2.0
```

This technique makes errors much easier to find.

2.3 Exercise 2

Use assignment to help you compute

$$3 - \frac{3^2 - 2 \cdot 3}{2 \cdot 3 - 2}$$

[6]:
```
# Type and run your code here.
```

3 Order of Operations

What is

$$5–2/6?$$

Is it $5 - 1/3$ or 3 divided by 6? That is, is it $5 - (2/6)$, or is it $(5 - 2)/6$? We should know that the first is true because division happens before subtraction. What makes knowing this necessary is because there are two operations above, subtraction and division, and the answer will depend on which is done first. The order of operations act like a traffic cop directing the flow. MATLAB uses a fairly standard order of priority (precedence) for operations: addition and subtraction have the same priority, which is below multiplication and division. Powers have the highest priority. So typing the command

[7]:
```
5 - 2/6
```

[7]:
```
4.666666666666667
```

returns an answer of 4.6667 or $5 - 1/3$, as division is done before the subtraction.

In the following exercise you will review the basic order of operations.

3.1 3.1 Multiplication/Division vs. Addition/Subtraction

3.2 Exercise 3

One of these things is not like the others. Which of these MATLAB commands is not like the other two? To help you out, try doing this with some values for a, b, and c, such as, a=3, b=13, c=23.

a. **(Q3)** Circle one:

1. `a - b*c`
2. `a - (b*c)`
3. `(a - b)*c`

b. **(Q4)** Circle one:

1. `a*(b - c)`
2. `(a*b) - c`
3. `a*b - c`

c. **(Q5)** Circle one:

1. `a/b + c`
2. `a/(b + c)`
3. `(a/b) + c`

d. **(Q6)** Circle one:

1. `(a + b)/c`
2. `a + (b/c)`
3. `a + b/c`

e. **(Q7)** Which operations have higher precedence? Circle one:

1. multiplication/division
2. addition/subtraction

[8]:
```
# Type your answer in a comment here
```

3.3 Multiplication/Division vs. Exponentiation

3.4 Exercise 4

Repeat the same exercise with the following expressions.

a. **(Q8)** Circle one:

1. `a**(b*c)`
2. `(a**b)*c`
3. `a**b*c`

b. **(Q9)** Circle one:

1. `a*(b**c)`
2. `a*b**c`
3. `(a*b)**c`

c. **(Q10)** Circle one:

1. `a/b**c`
2. `(a/b)**c`
3. `a/(b**c)`

d. **(Q11)** Circle one:

1. `a**b/c`
2. `(a**b)/c`
3. `a**(b/c)`

e. **(Q12)** Which operations have higher precedence? Circle one:

1. multiplication/division
2. exponentiation

f. **(Q13)** Using the rules you found, evaluate the following expression and rewrite it with parentheses showing the order in which MATLAB evaluates the operations. For example, `a + b + c` $= (a+b)+c$ and not $a+(b+c)$. In this example there are 4 operations, so your answer should have 3 pairs of parentheses.

[9]:
```
7 - 3**2/9 + 4
```

[9]:
```
10.0
```

[10]:
```
# Type your answer in a comment here
```

3.5 Operations with the same precedence

How does MATLAB interpret the following commands?

[11]:
```
3 - 3 - 3
```

[11]:
```
-3
```

[12]:
```
6/3/2
```

[12]:
```
1.0
```

[13]:
```
2**3**2
```

[13]:
```
512
```

There is an ambiguity as the answers are different if the leftmost operation is done first compared to if the rightmost one is.

3.6 Exercise 5

Evaluate the following expressions first using **left-to-right order**, and then from **right-to-left**. Enter both answers separated by a comma. 1. **(Q14)** 3 − 3 − 3 1. **(Q15)** 6 / 3 / 2 1. **(Q16)** 2 ** 3 ** 2

3.7 Exercise 6

Investigate the expressions below to see the order in which the operations are performed by MATLAB. (For example, does `5 - 3 - 2` $= (5-3)-2$ or does it equal $5-(3-2)$?)

[14]:
```
c=11
a=1
b=103
c=3
```

```
5 - 7 - 8
c - a + b - c
5/7/8
5/7*8*9
```

[14]:
```
51.42857142857143
```

[ ]:
```

```

(Q17) What rule(s) does MATLAB use when evaluating expressions with two or more operations of the same priority? Circle one: 1. from right to left 1. from middle to end 1. from left to right

3.8 Exercise 7

Practice what you have just reviewed to evaluate the following. Let

[15]:
```
a=4
b=5
c=8
```

a. **(Q18)**

$$\frac{a^b - c/b}{c - a}$$

b. **(Q19)**

$$\frac{a^{c-b}}{c - b}$$

c. **(Q20)**

$$\frac{a^{3/2}}{b}$$

d. **(Q21)**

$$\frac{a - b(c - a)}{c - a}$$

Note: A full explanation of MATLAB's priority rules is attached to this project in the Reference Section. You may want to review this to check your answers.

3.9 Reusing previous commands

You can save some typing if you learn how to reuse and edit your previously entered commands. The Command History window shows the commands you've typed during a MATLAB session. Double clicking on a command will paste it into the command window. The command history may also be accessed inside the command line by using either the up or down arrow to scroll through your previous commands.

Once a command is at the >> prompt, you may make changes to it. Use the left or right arrows to move around, or the mouse. Then you may insert text or delete existing text.

Finally, you may type (Shift+Enter) to execute the new command. You do not need to have the cursor at the end of the line to do this.

4 Function evaluation

MATLAB allows much more than just the basic arithmetic operations. Extra functionality is provided by functions, such as sin, cos, or sqrt. A function is referred to by its name and used by calling the function with its argument(s) enclosed in matching parentheses. A list of basic functions and their MATLAB equivalents is attached to the end of the project in the Reference Section.

For instance, the value of $\sqrt{15}$ is given by

[16]:
```
from numpy import *
sqrt(15)
```

[16]:
```
3.872983346207417
```

(The text after and including the number sign symbol is a comment and will be ignored if typed in.)

Just typing the function name, without the parentheses, will show its definition.

Here are a few more examples of function evaluations. Note that to get $\pi = 3.1415\ldots$, you type `pi`; to get the number $e = e^1 = 2.7182\ldots$, you type `exp(1)`.

When trying these examples, if you get an error message, check that you have spelled the function name correctly. For example, if you type `sqt(3)` (you misspelled `sqrt`), MATLAB responds with an error message: ——————————————————————————— NameError Traceback (most recent call last) <ipython-input-17-a2f0de13eb7d> in <module> —-> 1 sqt(3)

NameError: name 'sqt' is not defined Try evaluating the following using MATLAB:

[17]:
```
sqrt(216)
pi                      # pi = 3.1416... is built-in
exp(1)                  #  e = 2.7183... is not
sin(pi/4)
x = pi/3                # assigns x
tan(x)                  # returns 1.7321, as x = pi/3
sin(x)**2 + cos(x)**2 # returns 1, again, as x = pi/3
```

[17]:
```
1.0
```

Note: you will get better precision if you do not round off intermediate computations. Try typing the following to see an example:

[18]:
```
pi/4
sin(0.785)
sin(pi/4)
sqrt(2)/2
```

[18]:
```
0.7071067811865476
```

When trigonometric functions are evaluated in MATLAB, arguments must be specified in radian measure—not in degrees. Recall, to convert from degrees to radian you multiply by $\pi/180$.

4.1 Exercise 8

Use MATLAB to evaluate the following expressions.

a. **(Q26)** Calculate the sine of 40 degrees using MATLAB. MATLAB uses radians for all angle measurements. You will need to convert degrees to radians first.

b. **(Q27)** Evaluate $\sin^2 65°$

c. **(Q28)** Evaluate $e^{(10-8.5)/3}$

d. **(Q29)** Evaluate $\arcsin(\sin(3\pi/4))$

5 ~~Vectors~~ Arrays

We will often want to apply a function to many different values of x. We'd like to do this in the most convenient manner. For example, to compute the function $f(x) = x^2 \cos^3 x$ for $x == \pi/3, \pi/4, \pi/6$, we can save some work by assigning a value to x and then computing:

[19]:
```
x = pi/3
x**2 * cos(x)**3
```

[19]:
```
0.13707783890401892
```

[20]:
```
x = pi/4
x**2 * cos(x)**3
```

[20]:
```
0.218089506238715
```

[21]:
```
x = pi/6
x**2 * cos(x)**3
```

[21]:
```
0.17806933618012677
```

This allows us to make a single change to x per line, instead of changing it in both places.

Although the above technique is useful, there are better ways to do this task, as the MATLAB language is written to naturally apply the same function to many different values at once. In order to do so we need to learn two things: 1. How to store more than one number into a variable (vectors) 1. How to apply a function to all the values of the vector simultaneously.

We use the term vector to describe a MATLAB variable that contains lots of numbers at once. Vectors are made in MATLAB using the square brackets `[]`. (MATLAB refers to vectors as arrays, a more general concept.) The simplest way to make a vector is to just type in the numbers you want inside of matching `[]`: (Don't type the part with the `#` symbol. This is a comment to you.)

```
x = [1,1,2,3,5,8,13,21]      # start of Fibonacci sequence
x = [1 1 2 3 5 8 13 21]      # commas are optional
somePrimes = [2,3,5,7,11,13,17,19,23]
```

5.1 Exercise 9

Store the number 1, 2, 3 in a vector named x. Answer the following for this vector.

a. **(Q30)** What is `x + x`? Circle one:

1. The vector [1, 4, 9]
2. The vector [2, 4, 6]
3. An error

b. **(Q31)** What is the output of `x*x`? Circle one:

1. The vector [1, 4, 9]
2. The vector [2, 4, 6]
3. An error

5.2 Arithmetic sequences

Many of the vectors of numbers we will deal with will be arithmetic sequences:

$$a, a+h, a+2h, a+3h, ..., a+kh = b, \quad h > 0.$$

We think of this in two ways: 1. Numbers between a and b separated by a step size of h 1. A certain number (k + 1) of numbers evenly spaced between a and b.

There are two different ways in MATLAB to generate such sequences, depending on how you think of the values in the sequence.

5.2.1 Numbers separated by a step size h

To generate a sequence of numbers separated by 1 is done using the `:` symbol, as in `a:b`:

[22]:
```
r_[1:5]
```

[22]:
```
array([1, 2, 3, 4])
```

[23]:
```
r_[-1:5]
```

[23]:
```
array([-1,  0,  1,  2,  3,  4])
```

[24]:
```
-r_[1:5]
```

[24]:
```
array([-1, -2, -3, -4])
```

The last example shows that the minus sign here means multiply each entry by -1.

If we want a step size of h we use this syntax: `a:h:b`. For instance:

[25]:
```
r_[0:10:2], r_[0:9:2], r_[0:8:2]
```

[25]:
```
(array([0, 2, 4, 6, 8]), array([0, 2, 4, 6, 8]), array([0, 2, 4, 6]))
```

[26]:
```
r_[0:60:15]
```

[26]:
```
array([ 0, 15, 30, 45])
```

[27]:
```
15*r_[0:4]
```

[27]:
```
array([ 0, 15, 30, 45])
```

5.3 Exercise 10

Find MATLAB commands which generate the following lists. Make sure your answer is correct.

a. **(Q32)** The odd numbers 1, 3, ..., 99

b. **(Q33)** The numbers 10, 20, 30, ..., 120

5.4 A fixed number of numbers

When plotting functions we will desire a lot (say 100 or a 1000) of evenly-spaced numbers between two points. Rather than figure out the step size between the points it is more convenient to specify how many linearly spaced points we want. This is done with the linspace function. Using it as `linspace(a,b)` will produce 100 numbers between a and b. You can override the default of ~~100~~ 50 using a third argument:

[28]:
```
linspace(0,5,6)
```

[28]:
```
array([0., 1., 2., 3., 4., 5.])
```

[29]:
```
linspace(0,pi,3)
```

[29]:
```
array([0.        , 1.57079633, 3.14159265])
```

5.5 Exercise 11

a. **(Q34)** What linspace command produces this output:

array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4.])

Circle one:

1. linspace(1,7,4)
2. linspace(1,4,1/2)
3. linspace(1,4,7)
4. linspace(1,1/2,4)

b. **(Q35)** What is the last value output by the command

linspace(0,pi)

[30]:
```
linspace(1,4,7)
```

[30]:
```
array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. ])
```

5.6 Applying arithmetic operations to vectors

For this project, we focus on arithmetic operations which work as we would like: addition of vectors of the same size, and multiplication of a vector times a number (a scalar).

For example:

[31]:
```
x = r_[1:6]
x
```

[31]:
```
array([1, 2, 3, 4, 5])
```

[32]:
```
x + x
```

[32]:
```
array([ 2,  4,  6,  8, 10])
```

[33]:
```
3*x
```

[33]:
```
array([ 3,  6,  9, 12, 15])
```

[34]:
```
3*(x - 3)
```

[34]:
```
array([-6, -3,  0,  3,  6])
```

This allows us to do simple transformations, such as the conversion from Celsius to Fahrenheit, or back, given by these formulas:

$$F = 9/5C + 32 \quad \text{or} \quad C = 5/9(F-32)$$

This is the basis of the last exercise.

5.7 Exercise 12

Two common measures of temperature are the Fahrenheit and the Centigrade scales.

a. **(Q36)** Find room temperature in Celsius ($F = 68°$)

b. **(Q37)** Find the average body temperature in Celsius ($F = 98.6°$)

c. **(Q38)** Let X be a vector of Fahrenheit values between -100 and 100 in step size of 20, and Y be the corresponding Celsius values. Which MATLAB commands give this? Circle one:

1. `X = r_[-100:110:10]; Y = 9/5*X + 32; Y`
2. `X = r_[-100:120:20]; Y = 9/5*X + 32; Y`
3. `X = r_[-100:110:10]; Y = 5/9*(X - 32); Y`
4. `X = r_[-100:120:20]; Y = 5/9*(X - 32); Y`

[35]:
```
X = r_[-100:110:10]; Y = 9/5*X + 32; Y
X = r_[-100:120:20]; Y = 9/5*X + 32; Y
X = r_[-100:110:10]; Y = 5/9*(X - 32); Y
```

```
X = r_[-100:120:20]; Y = 5/9*(X - 32); Y
```

[35]:
```
array([-73.33333333, -62.22222222, -51.11111111, -40.        ,
       -28.88888889, -17.77777778,  -6.66666667,   4.44444444,
        15.55555556,  26.66666667,  37.77777778])
```

d. **(Q39)** If two vectors are the same length, a table can be made from them that allows you to compare their entries. The syntax is either `[X;Y]` or `[X;Y]'`. Look carefully at the table of X and Y values. At what temperature is the Celsius and Fahrenheit measurement the same?

[36]:
```
X = r_[-100:120:20]
Y = 5/9*(X - 32)
c_[X,Y]
```

[36]:
```
array([[-100.        ,  -73.33333333],
       [ -80.        ,  -62.22222222],
       [ -60.        ,  -51.11111111],
       [ -40.        ,  -40.        ],
       [ -20.        ,  -28.88888889],
       [   0.        ,  -17.77777778],
       [  20.        ,   -6.66666667],
       [  40.        ,    4.44444444],
       [  60.        ,   15.55555556],
       [  80.        ,   26.66666667],
       [ 100.        ,   37.77777778]])
```

6 Reference Section

6.1 Basic MATLAB commands

Description	Symbol(s)	Comments
Number (scalar) assignment	+	
Scalar arithmetic		
Vector assignment		
Scalar/vector operations		
Pointwise vector operations		
Separator	;	
Table		

6.2 Order of Operations

MATLAB uses the following symbols for arithmetic operations

Operation	Symbol	Precedence	Comment
Exponentiation	**	3	Highest precedence
Multiplication	*	2	

Operation	Symbol	Precedence	Comment
Division	/	2	
Addition	+	1	Lowest precedence
Subtraction	-	1	Lowest precedence

If an arithmetic expression contains nested parentheses, then the expressions contained within the innermost parentheses are evaluated first. In the absence of parentheses, the precedence rules decide the order of evaluation. The following rules apply: 1. All operations with a higher precedence are carried out before those of lower precedences. Thus exponentiation is carried out first, then comes multiplication and division, and finally, addi- tion and subtraction. 1. If two operations have the same precedence then the operation on the left is carried out first. This is called the left-to-right scan rule. The examples in the following table show how the precedence rules help interpret arithmetic expressions based on these rules. Try them out on the computer and make sure that you understand how each expression is interpreted.

MATLAB notation	Mathematical notation	Meaning of the operation
`sqrt(x)`		
`abs(x)`		
`sign(x)`		
`exp(x)`		
`log(x)`		
`log10(x)`		
`sin(x)`		
`cos(x)`		
`tan(x)`		
`arcsin(x)`		
`arccos(x)`		
`arctan(x)`		

[ ]:

NOTE: MATLAB may give unexpected results when working with negative numbers, since its default is to assume inputs and outputs are complex numbers. Try evaluating `(-4)**(1/2)`, `sqrt(-4)`, `(-8)**(1/3)`, and `log(-5)`, to see what happens. Note that `1j` represents $\sqrt{-1}$

[37]:
```
1j
```

[37]:
```
1j
```

The `#` and beyond are comments and need not have been typed in.

*Recall from **Lab 1** that we run a cell in Jupyter Lab by pressing SHIFT+ENTER. Occationally you may have to run a cell twice to get the plot to show.*

[ ]: