MATLAB Primer

Preliminaries

- MATLAB is short for Matrix Laboratory
- When you open MATLAB - there should be a command window, workspace, command history, and Editor. Some of these can be undocked from the main screen. If you do not see all of these screens, click on home in the upper left hand corner and then click on Layout on the top bar towards the right.
- The command window is where you will view the output of your code. You could also type some input here.
- The workspace shows you all of your current working variables and other objects.
- The current directory lets you know where you are and where files will be saved to. You can click on the folder with the arrows to go up or down a directory/folder to get in the folder you are looking for.
- The Editor is where you will write and edit m files. To run or execute the m file, you click on the Run button (green play button) on the toolbar at the top (need to be on the Editor tab).
- MATLAB is case sensitive, $x \neq X$
- Abort / kill a job: click on a MATLAB scree and then do CTRL+C. Do this if your program is not doing what it should and you need to stop it.
- Saving an m-file: Do not put any spaces in the name of your m-file.
- All m-files should have `close all` and `clear all` at the top. This clears out the variables.
- MATLAB **Help**: Google is your best friend. There are many pages devoted to describing different commands. If there is something that you think MATLAB should be able to do, it can probably do it. In addition to Google, you can use the help command. For example, if you type the following into the command window:

  ```
  help sqrt
  ```

  You will then get the following in the command window:

  ```
  sqrt   Square root.
      sqrt(X) is the square root of the elements of X. Complex
      results are produced if X is not positive.

      See also sqrtm, realsqrt, hypot.

      Overloaded methods:
         codistributed/sqrt
         gpuArray/sqrt
         sym/sqrt

      Reference page in Help browser
         doc sqrt
  ```

  If any of these appear in blue, you can click on them to get more information.
- MATLAB as a **Scientific Calculator**: The following command,

  ```
  exp(5)*sin(pi/4)
  ```

  will be evaluated where `exp` is the exponential function, `sin` is the sine function, and `pi` is π. MAtLAB can be used as a calculator and uses the standard arithmetic operators: `+, -` $\times$, $\div$, and $\wedge$ for powers.

Arrays

Look at:

`ExampleTutorial_Arrays.m`

Most variables or parameters will be stored as arrays or matrices. The command

```
x=[1 2 3]
```

will store x as the vector $x(1) = 1$, $x(2) = 2$, $x(3) = 3$. If you type this into MATLAB in the command window or create an m file with this and run it, you will see the following output in the Command Window:

```
x =

     1     2     3
```

To suppress the output, use a semi colon after each expression. A few useful commands:

- `length` and `size`: For a vector, `length` will give you the number of entries in a vector. For a matrix or vector, `size` will give you the number of rows and the number of columns.
- Creating a vector of evenly spaced entries: you can do it with

  ```
  x=[0:0.1:2]
  ```

  if you want the vector to start at 0, end at 2, and increment by 0.1. So, x(1)=0, x(2)=0.1, ... or you can use the linspace command with `x=linspace(0,2,21)` to start at 0, end at 2, and increment by whatever is necessary to have exactly 21 entries in the vector.
- `eye(n)`: creates an n×n identity matrix
- `zeros(m,n)`: creates a m×n matrix of zeros
- `ones(m,n)`: creates a m×n matrix of ones
- scalar multiply: multiplication entry by entry of vectors uses `.*`
- scalar divide: division entry by entry of vectors uses `./`
- element by element exponentiation of a vector uses `.∧`
- matrix divide: uses `/`
- transpose of a matrix or vector: use `'`

Conditional Statements, For Loops, and While Loops

Look at the following for examples on Conditional Statements:

`ExampleTutorial_Conditionals.m`

The execution of an if or while loop depends upon the evaluation of a condition. To construct conditions, MATLAB provides six relational operations:

- < (less than)
- <= (less than or equal to)
- > (greater than)
- >= (greater than or equal to)

- == (equal to)
- ∼= (not equal to)

and three logical operators

- & (and)
- | (or)
- ∼ (not)

Note that a single equal sign (=) is used to assign a value to a variable such as $a = 4$ whereas the double equal sign (==) compares two arguments.

An if statement allows a statement to be evaluated only if some condition is true. This is called a branch statement. Typically, if you have an if statement, and if it is not satisfied, a corresponding else or elseif statement can be given. The simplest form of an if statement is (in pseudocode):

```
if(condition)
       evaluate statements
end
```

For and While Loops: A for loop is used to repeat a statement or a group of statements for a fixed number of times. A while loop is repeatedly executed while a statement or group of statements remains true. For example:

```
     for m=1:100
      num(m)=1/(m+1)
end
```

The counter increments automatically by 1 in this statement and will print out the value of 1/(m+1) for $1 \leq m \leq 100$ since there is not a semicolon after the num statement in the for loop. The counter in the loop can also be given an explicit increment, such as `i=m:k:n`, to start the counter i at m and advance the counter by increments of k until reaching n.

Look at the following for examples on Conditional Statements:

`ExampleTutorial_Conditionals.m`

Plots

Look at the following on canvas for examples on plots:

`functionapproxex.m`

You can plot a function using the plot command.

Taylor Series and Error

Question: What is the minimal value of N required for:

$$|ln(1.5) - P_N(1.5)| < 10^{-5}$$

where P_N is the N^{th} term in the Taylor polynomial for $f(x) = ln(x)$.

Recall, the N^{th} order Taylor expansion of $f(x)$ at $x = x_o$ is:

$$P_N(x) = \sum_{k=0}^{N} \frac{f^{(k)}(x_o)}{k!}(x - x_o)^k$$

The Nth order Taylor polynomial for $f(x) = ln(x)$ about $x_o = 1$ can be found by evaluating the function and its derivatives at $x = 1$: $f^0(1) = \ln(1) = 0$, $f'(1) = (1/x)|_{x=1} = 1$, $f''(1) = (-1/x^2)|_{x=1} = -1$, $f^{(3)}(1) = (2/x^3)|_{x=1} = 2$, ... $f^{(n)} = (n-1)!(-1)^{n+1}$. We have:

$$P_N(x) = 0 + 1(x-1) - (x-1)^2 + \frac{2!}{3!}(x-1)^3 - \frac{3!}{4!} + (x-1)^4 + \cdots$$

We can write this compactly as:

$$P_N(x) = \sum_{k=1}^{N} \frac{(-1)^{k+1}(x-1)^k}{k}$$

Also, the error for this alternating series is:

$$|ln(1.5) - P_N(1.5)| \leq |P_{N+1}|$$

Pseudocode: The idea is to keep adding a new term to series and check if $|\ln(1.5) - P_N(1.5)| < 10^{-5}$. Overview of code: INPUTs that we need to declare at the beginning: value x=1.5, tolerance TOL=10^{-5}, maximum number of iterations M. OUTPUTs that we are looking for is the number of terms N or a message of failure.

```
STEP 1  - initialize
TOL=1e-5;
SUM=0;
N=1;
X=1.5;
Y=X-1;
M=1000;
STEP 2 While N<=M, do steps 3-5
       STEP 3
              Set SUM as follows: SUM=SUM+((Y^N)*(-1)^(N+1))/N;
      Check if meet criteria: check=abs(log(X)-SUM);
 STEP 4 If |SUM|<TOL then
              OUTPUT(N)
                   STOP
       STEP 5 Set N=N+1
STEP 6 OUTPUT('Method Failed');
STOP
```

Code in MATLAB:

```
close all;
clear all;

TOL=1e-5;
SUM=0;
N=1;
X=1.5;
Y=X-1;
M=1000;

while(N<=M)
    SUM=SUM+((Y^N)*(-1)^(N+1))/N;
    check=abs(log(X)-SUM);
    if(check<TOL)
        display(N)
        break
    end
    N=N+1;
end
```