Algorithms and Script Files

Leigh J. Little SUNY Brockport September 2, 2022

# Contents



# 1 Algorithms

An algorithm is a set of steps designed to accomplish a task. You have been using algorithms all your life. For example, most of you have a set series of steps you carry out each morning when you get up and get ready for the day. This is an algorithm.

A computer program is an algorithm. The most important thing to remember about algorithms as they apply to computer programs is that programs are dumb. They do exactly what you tell them to. It turns out that it is very easy to tell a computer program to do something different than what you intended to tell it.

Another example of an algorithm you have all seen is

```
Lather
Rinse
Repeat
```

There is a problem with this algorithm if you were to have a computer execute it. There is no way to end the algorithm. As humans, we insert additional steps into the process because we don't interpret the steps in a literal sense. A computer is not capable of this. It will reach the last step of the and then repeat the process. It will keep doing this forever because there is no way for it to break out of the cycle. This is an example of what is called an infinite loop. Infinite loops in computer programs are bad (most of the time).

As another example, consider the algorithm below:

```
Brush teeth
Drink orange juice
```

There is nothing technically wrong with this algorithm, but the sequence of steps is likely to have unpleasant consequences.

Over time we will make some observations about computer programs

a) A computer program begins with a statement of what the program needs to accomplish. This is called the problem statement.

b) This statement is then turned into an series of calculations necessary to accomplish the stated goal.

c) Some of the steps need to have a specific sequencing in order for the algorithm to succeed.

d) Some steps can happen anywhere during the calculation. It doesn't matter when they are done as long as they are performed at some point.

e) Some steps are completely optional. The don't need to be performed, but they might make other steps easier if they are done.

# 2 Script Files

Suppose you had a calculation you were performing in MATLAB. It has a total of 57 steps. You are on step 36 when you realize you made an error on step 12. You can re-enter the correct step 12 (maybe) and use the up-arrow key to bring back previous calculations so that you can save some time, but this is also likely to be error prone. You might skip step 24 by accident.

Working directly from the command line is useful for simple operations, but for long sequences of calculations, a more reliable process is needed. The answer to this is a *script file*. A script file is a plain text file containing a sequence of MATLAB commands. You can enter your commands into the file, then execute (or run) the file. MATLAB will perform the operations in the file. The benefit of this is that if you discover an error in your calculation, you can correct the error in the script file, then re-run the script.

Here are the general guidelines for using script files:

- A script file must end with a `.m` extension. Note that this is the same extension used by Mathematica.
- You need to choose a name for your script file. This must not be the same as an existing MATLAB command, variable name or another script file that is in your current working directory.
- The script file must be in the same working directory that MATLAB is in (this is not 100% correct, but we will do this for now).
- To run the script file, you type in the name of the script file at the command line (minus the `.m` extension). For example, if you created a script called `myscript.m`, you would type

  ```
  >> myscript
  ```

  to run the script.

## 2.1 The `input` Command

The ability for a script to ask the user to input values is useful. This can be accomplished with the `input` command. The syntax for this command is

```
>> x = input('Input a value for x ');
```

The script will stop execution and ask the user to input a value for `x`, then proceed with the remainder of the steps. You should always include a text prompt to indicate to the user that the script is waiting for input.

## 2.2 Commenting Your Script

The need to include comments in your scripts frequently occurs. These comments can consist of statements indicating what the script is doing, how it is performing certain algorithms or other auxiliary information. In MATLAB, the percent sign (`%`) is used to indicate a comment. Any text that occurs after a `%` is ignored by MATLAB.

## 2.3 A Short Example

We will be writing many scripts throughout the semester. A short example is given below.

```
% A short script to compute a quadratic expression for some
% user-input value
%
clear   % Trailing comments are possible
        % The clear command ensures the script is starting from a blank
        % environment
x = input('Input x: ');
y = 2*x^2 - 3*x + 6
```

Enter the text above into the MATLAB file editor and save it as `ex1.m`. You can then execute the script as follows:

```
>> ex1
Input x: 3
y =
    15
```

Notice that there is no `.m` in the command to execute the script. Also, the semicolon was left off the expression for `y` so that the value is copied to the screen.

### 2.4 Stopping a Script

If you need to stop a script from executing, you can press `CTRL-C`. This is useful if you realize you made a mistake or if you think the script might have gone into an infinite loop.

## 3 Programming Errors

Unfortunately, programming is rarely perfect on the first try. Errors can and do occur. Programming errors are broken down into two categories.

### 3.1 Syntax Errors

Syntax errors occur when you type in a command that MATLAB cannot understand. An example of this is not having the same number of left parentheses as right parentheses in an expression. Suppose you wanted to compute

$$z = \sin\left((x+y) - \left(x^2+y^2\right)\right)$$

and you entered

```
>>  z = sin((x+y) - (x^2+y^2)
```

The expression above has three left parentheses, but only two right parentheses. MATLAB will generate an the error

```
>>  z = sin((x+y) - (x^2+y^2)
  z = sin((x+y) - (x^2+y^2)

Error: Invalid expression. When calling a function or indexing a variable, use
parentheses.  Otherwise, check for mismatched delimiters.
```

Syntax errors happen all the time even for experienced programmers but they are usually easy to diagnose. MATLAB is more than happy to let you know that you have one.

### 3.2 Logic Errors

A logic error occurs when you tell your program to do something other than what you intended. Logic errors are much more difficult to diagnose. This is because computer programs do not obey what is called the Law of Local Cause and Effect. This law simply means that the source of an error is usually near the location where the error appears. For example, if you are driving down the road and large clouds of smoke appear from underneath the hood of your car, there is likely something wrong with your engine or cooling system.

In a computer program, an error can become apparent far from where the error was made. As a (not very good) analogy, suppose smoke starts coming from under the hood of your computer program. You open the hood and everything looks fine. After examining the situation for a long time, you finally discover that the problem is that you don't have enough air in your tires.

Drawing from the example in the previous section, suppose you were computing

$$z = \sin\left((x+y) - \left(x^2+y^2\right)\right)$$

and you entered

```
>>  z = sin((x+y) - (x^2) + y^2)
```

In this case the number of left and right parentheses match and MATLAB will compute a value for this. However, the parentheses are in the wrong place and the value for $z$ will not be correct. The error in the formula will not become apparent until later in the calculation. The frustrating aspect of logic errors is that 'later' can be several hundreds lines of calculations later.