ANALYSIS OF MATLAB

A PREPRINT

Lee, Evan **Tu, Sebastian**

April 12, 2019

ABSTRACT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

***K*eywords** First keyword · Second keyword · More

1 Introduction

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

2 History

MATLAB's original purpose was to be a strong matrix calculator, and nothing more. [1] Algorithms for solving matrix linear equations and eigenvalue programs were written in a research paper by J. H. Wilkinson and eighteen of his colleagues. MATLAB was born out of Fortran, with matrices as the only data type. In 1983, MATLAB creator Cleve Moler's associate Jack Little used a Compaq PC clone to write a more extended version of MATLAB. Functions, toolboxes, graphics, and many other additions were made to MATLAB with this new version. What resulted was named PC-MATLAB, and was revealed at the IEEE Conference on Decision and Control in December of 1984. Since then, MATLAB has continued to develop uses and capabilities for technical computing. Computation, visualization, and programming are its key components. Today, it still has a presence as a private company which specializes in mathematical computation. Matlab today is still used due to its graphical displays and "toolboxes", which greatly expand the language to interact with other languages and programs such as Microsoft Excel and R. [3]

3 Control Structures

While MATLAB has plenty of control structures, it does not contain any special or unique control structures as compared to other languages such as Ruby's Until loop (which is a while not loop). MATLAB's control structures can be split into two categories: branching statements and loop statements.[2]

3.1 Branch Statements

Branch statements compare logical conditions to decide which program block should be executed. MATLAB has two types of branch statements: if conditions and switch statements.

Branch statements in MATLAB consist of if statements and switch cases. An if statement in MATLAB is written as

```
if (1 == 1)
    display("this will be shown");
else
    display("this will not be shown");
```

In the example above, the code uses an ELSE clause, which handles cases that evaluate to false in the if statement. If statements can be used together to create nested-if statements, which allow for more complex conditions. An example of this is :

```
if (1 == 1)
    if (2 == 2)
        display("This will pass through two conditions");
    else
        display("This else is part of the inner if statement);
else
    display("This else is part of the outer if statement");
```

Switch statements are similar to if statements,though they allow for multiple branches, instead of a single branch in the case of an if statement. They have a variable which is used as the "switch", and will execute a specific code block based on the value of the switch. An example of a switch statement is:

```
testScore = 'A'
switch (testScore)
    case 'A'
        display("Good Job!");
    case 'B'
        display("Nice!");
    case 'C'
        display("C's get degrees!");
    case 'D'
        display("Close!");
```

In this case, the code block that will be executed is case 'A', which will output the string "Good Job!" to the standard output.

3.2 Loop Statements

Loop statements are used to run a block of code a certain number of times. MATLAB has two types of loops statements: while loops and for loops.

While loops run a block of code continuously until their condition is broken. They effectively are repeatedly calling an if statement and running the code inside as long the condition evaluates to true. The MATLAB implementation of this is:

```
count = 0;
while (count < 5){
display ("This portion will be printed out 5 times);
count = count + 1;
}
```

In the example above, the while loop checks a variable's value (count). The code block repeats as many times as the the condition holds true, so in this case, 5 times.

For loops are similar to while loops, with more information built in. For loops have 3 portions: a block that will only be executed once at the beginning of the loop, a conditional block for the loop, and a block that is executed after every pass through the loop. MATLAB implements for loops as such:

```
for(i=0; i < 5; i= i + 1){
    display("This will be printed out 5 times");
}
```

The example above uses the first portion of the loop to create a variable i, the second portion to check the value of i, and the third portion to increment i by one. This effectively does the same as the while loop example above.

4 Data Types

4.1 Integer, Unsigned Integer, Floating Point

Integers, floating point, and unsigned integers are stored by default as double-precision floating point numbers. They can be specified as single-precision numbers, which offer much more memory-efficient storage than double-precision numbers. You can expose and output properties and traits of the type by using the command whos. Integers will be represented by the class double.

Input:

```
X = 10;
whos x;
```

Output:

```
Name        Size              Bytes  Class     Attributes

x           1x1               8      double
```

4.2 Character Array and String Array

In MatLab, strings are representations of character arrays. Just like any generic array, you can make arrays of strings, which will be discussed alongside all other data structures further on. Using the String function, you can convert an arbitrary data type into a string array using the string function.

Input:

```
str = "Hello, world"
str = ["Mercury","Gemini","Apollo"; "Skylab","Skylab B","ISS"]
```

Output:

```
str = 2x3 string array
   "Mercury"    "Gemini"      "Apollo"
   "Skylab"     "Skylab B"    "ISS"
```

Input:

```
str = ["hello", "world"];
x = join (str, " ");
disp(x);
```

Output:

```
"hello world"
```

4.3 Numeric Array

Any numeric value can be stored as single or double precision arrays. There are functions (ex: int8) for specifying byte sizes: including 8, 16, 32, 64; along with equivalent functions for unsigned integers (ex: uint8). There are also functions for the testing of numerous array contents, including: integer, float, numeric, real, finite, infinite, and NaN.

Input:

```
w = int8(10);
x = int8(11);
disp(isinteger(w));
disp(isfloat(x));
```

Output:

```
1
0
```

Seeing as MatLab stands for "Matrix Laboratory", all arrays are two dimensional arrays, or matrices. To create these, you separate horizontal elements by whitespace and rows by semicolons.

Input:

```
a = [1  2  3  4]
```

Output:

```
a = 1x4
  1     2     3     4
```

5 Subprograms

Subprograms (also known as procedures, subroutines, or functions) are an integeral portion of any programming language.[2] Similar to most programming languages, functions in MATLAB are defined with a name, return value, and list of parameters in the function declaration. As an interpreted language similar to Python, MATLAB does not need to specify the data types of the parameters or return value. Subroutines in MATLAB have two distinctive rules about function placements:

- In a function file which contains only function definitions. The name of the file should match the name of the first function in the file.
- In a script file which contains commands and function definitions. Functions must be at the end of the file. Script files cannot have the same name as a function in the file. [3]

Functions in MATLAB are declared with the function keyword followed by the return values. THis is followed by the = <function name>(<parameters>). The function scope ends by adding the keyword end to the end of the block code. Code documentation can be added as comments on the line after the function declaration. Here is an example of a simple function that returns the sum of two numbers

```
function x = sum(;,b)
% This will act as a help file.
% Everything that is written here that
% is part of this code block will be
% shown when using the command: help sum
    x = a + b
end
```

Input:

```
help sum
```

Output:

```
This will act as a help file.
Everything that is written here that
is part of this code block will be
shown when using the command: help sum
```

MATLAB allows for multiple values to be returned. To specify a multi-output function, wrap all the return values with brackets. An example of a multi-output function is shown below.

```
function [a,b] = clone(i)
    a = i;
    b = i;
end
```

In order to call functions in MATLAB, call the method in a script file. As mentioned before, functions can be placed either directly into a script file or stored in a funcion file.

Input:

```
disp(hello());
function str = hello()
    str = "Hello World";
end
```

Output:

```
"Hello World"
```

Functions can use other functions. This is called chaining. Functions in a function file that do not have the same name as the file can only be used by other functions within the file. This is known as a *local function* in MATLAB.

```
function a = func1()
    func2()
end

function b = func2()
    b = "This will be returned"
end
```

6 Summary

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

[**?**, **?**] and see [**?**].

References

[1] Gdeisat, Munther, Francis Lilley, and Ebrary, Inc. Matlab by Example : Programming Basics. Boston, Mass.: Elsevier, 2013. Elsevier Insights. Web.

[2] Tutorialspoint.com

`https://www.tutorialspoint.com/matlab/`

[3] MathWorks Documentation.

`https://www.mathworks.com/help/matlab/`

[4] MathWorks Forum.

https://www.mathworks.com/matlabcentral/answers/?status=answered&s_tid=alr_ban

[5] https://electronicsforu.com/electronics-projects/audio-compression-haar-wavelet-matlab
Audio Compression and Analysis. An audio signal sample is taken and analysed using MATLAB for frequency and amplitude. Haar and Daubenches algorithms are applied on the speech signal and the audio is compressed. Audio sizes before and after compression are compared.