

VG101: Introduction to Computer and Programming

Recitation Class Notes

Chenhao YE, Luotian Xu

VG101 TA Group, UM-SJTU Joint Institute

2018 Fall

Source code available at

<https://github.com/Ye-Chenhao/VG101-Recitation-Class-Notes>

Outline

1 About VG101

2 MATLAB

- About MATLAB
- Interface
- Syntax

1 About VG101

2 MATLAB

How to learn this course

How to learn this course?

- Follow the lecture
- Finish the homework on time

How to learn this course well?

- Don't do something stupid (including **violation of honor code**, late submission)
- Don't be afraid of asking and **exploring**
- Do as many labs as you can

Tips

- This course mainly focuses on the correctness of your code; you are not expected to improve the efficiency of code.
- Every **error** (mostly in red) means something is wrong. Fix it even when the outputs are sometimes right.
- Please be aware that a program which is right sometimes and wrong the other time, is **wrong**.

About honor code

First, please remember that you are **strongly encouraged** to discuss! Being brave to discuss with DALAO is an essential step to become a DALAO.

What you should not do?

- Don't show any code!
- Don't copy others' code!

Most of time, when you are violating honor code, you yourself know that you are doing something bad.

Please be aware that everytime you trying to violate the honor code but thinking that you can escape from the consequence, you are essentially challenging your instructor and TA group. All the code will be examined by specific tools, and almost all the tricks you can come up with are pale and useless.

Outline

1 About VG101

2 MATLAB

- About MATLAB
- Interface
- Syntax

About MATLAB

- Commercial and expensive (i.e. Octave)
- Not for general purpose (no one will write an operating system using MATLAB)
- Excel at numerical calculations and plot
- Many useful toolboxes (though we will not cover these in this course, you will use some of them in advanced courses)
- Widely used in academia

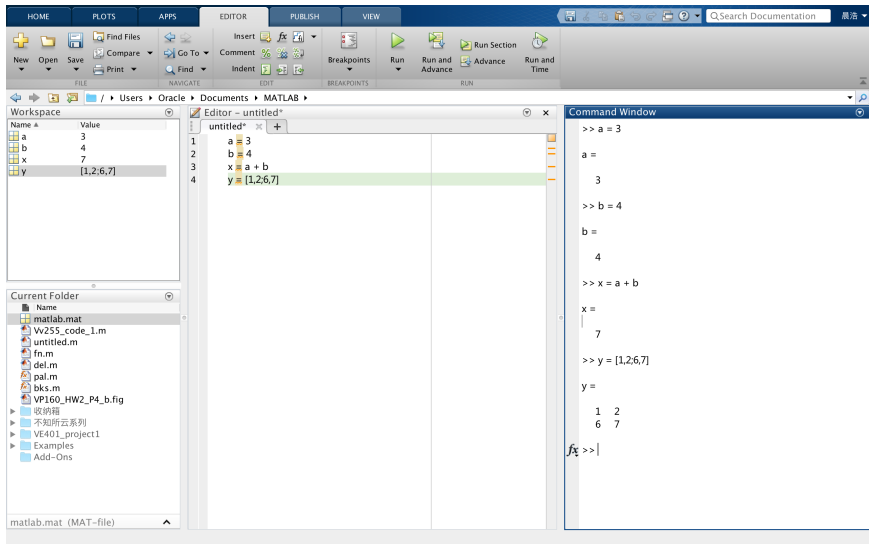
Outline

1 About VG101

2 MATLAB

- About MATLAB
- **Interface**
- Syntax

Interface



Interface

Workspace

Store and show what variables are available and their value. You can double-click a variable to see its details (useful for matrix). You can also save and load workspace.

Command Window

Input command direct into the command line. Variables can be seen in the workspace.

Editor

Editor is actually where you “buffer” your commands. **Run** the code in the editor is equivalent to type the command line by line into command window.

Interface

Current Folder (& Environment)

It indicates which folder you are currently. When you operate on file (e.g. call a function from another file), MATLAB will search the file in current directory or search path. Personally, I suggest managing your file in the current directory, which will make your life easier.

For more details about search path, you can refer to
https://ww2.mathworks.cn/help/matlab/matlab_env/what-is-the-matlab-search-path.html?lang=en.

Outline

1 About VG101

2 MATLAB

- About MATLAB
- Interface
- Syntax

Operators & Keys

- “+”, “-”, “*”, “/”, “.*”, “./”, “^”, “.^”: matrix or element-wise arithmetic
- “= ”: assignment
- “== ”: equal
- “&&”, “—— ”: logical and / or
- “% ”: comment. Will be ignored by MATLAB. Make the code easier to understand
- “; ”: suppress output. Please add it to every line that you do not expect output! Extra output may result in deduction
- “↑, ↓ ”: view history
- Tab: indent
- “clc ”: clear command window
- “clear / clear all ”: clear workspace

Variables

A valid variable name should

- Start with a letter (either upper or lower case)
- Followed by letters/digits/underscores
- Note that MATLAB is case sensitive

A good variable name should¹

- Have appropriate length (approximately 3 to 10 letters)
- Self explain its function
 - bad variable name: “a1”, “a2”, “a3”, “b”, “c”, “d”, “e”, “f”
 - good variable name: “counter”, “input”, “flag”

Iteration variables “i”, “j”, “k”

- Short, everyone use them

¹Not required by VG101 but a good habit.

Variables

MATLAB does not require variable declaration. When MATLAB encounters a new variable name, it automatically creates the variable and allocates memory for it. You don't need to worry about the type or size of the variable, MATLAB do all the things for you.

- This is good but if you misspell, you get in trouble
- Double click on a variable, and MATLAB will darken places it appears

Matrix & Vector & Array

Definition

Matrix: two-dimensional numeric arrays.

Vector: one-dimensional numeric arrays.

How to initialize array?

- $A = [1, 2, 3; 4, 5, 6; 7, 8, 9]$
- $A = \text{zeros}(3, 2)$ (row first, column next)

How to initialize vector?

- $A = [1:100]$ (step is 1 by default)
- $A = [0:0.01:1]$ (start, step, end)

MATLAB allows dynamic changes of array sizes. For example:

- $A = [A; A]$
- $A = [0, A] + [A, 0]$ (What can this do?)

Matrix & Vector & Array

How to access an element of a matrix?

- $A(2, 3)$ = The element at the second row and third column

How to access an element of a vector?

- $A(2)$ = The second element in A

How to cut a sub matrix?

- Use “:” ! It is the most useful symbol in MATLAB.
- $A(:, 1)$ = the first column of A
- $A(1, :)$ = the first row of A
- $A(1: 3, :)$ = the sub matrix consisting the first to third row of A
- $A(1: 3, 1: 3)$ = the upper-left corner 3×3 sub matrix of A

Matrix & Vector & Array

Aside: Matrix & Parallel Computation²

Why usually it's much faster to use matrix operation other than working with each element sequentially? This is mainly because of parallel computation. When a large problem can be divided into smaller tasks and they can be solved simultaneously, it can save a lot of time. Matrix is excellent for parallel computation since it can be divided into sub-matrices. Usually, matrix operation is 10 to 1000 times faster than iteration through elements.

²“Aside” means the content below will be discussed in more advanced courses, but it can be quite helpful if you know it now.

Aside: Array & Data Structure

Array is the first and most basic data structure you will met. Data structure is a data organization, management and storage format that enables efficient access and modification.³ You can think of array as cabinet in our bathroom in D22/D21. Once you remember the number on the cabinet (for array, indices), you can quickly access the content in it. Array, regardless its dimension, is stored as a 1D sequence in memory. This enables convenient memory management and quick access. In latter courses, you will encounter more data structures.

⁴Reference: https://en.wikipedia.org/wiki/Data_structure

Some Special Variables & Constants

Tips

All variables in MATLAB are matrix. Scalar is essentially a 1×1 matrix, but it allows some operations that may be invalid for a 1×1 matrix.

- “ans”: default output variable; do not use “ans” as your own variable name in your script
- “i”, “j”: virtual number
- “Inf”: infinite
- “NaN”: not a number; mostly appear when the result exceeds some limit or some errors occur
- “pi”

save & load

- save xxx: save all current variables into .mat file
- load xxx: load .mat file; will not clear variables already in workspace; will cover the variable in workspace if two variables share the same name

Command

- “exist”: test the existence of a variable or a file (exist xxx)
- “global”: declare global variable
- “help”, “doc”: show document (help xxx; doc xxx)
- “clc”: clear command window
- “clear / clear all”: clear workspace

It's good habit to add “clc” and “clear” at the first of your program, **especially in your homework.**

Control Flow

- if, while, for
- while 1

```
1  for i = 1:5
2      disp(i);
3      if i == 4
4          disp('look, it is a
           four!')
5      end
6  end
7  j = 100;
8  while j > 95
9      disp(j);
10     j = j - 1;
11 end
```

Outputs:

1 ←

2 ←

3 ←

4 ←

look, it is a four! ←

5 ←

100 ←

99 ←

98 ←

97 ←

96 ←

Control Flow

- break: break the loop
- continue: ignore the later code in this iteration; skip directly to the next iteration

```
1 i = 10;  
2 while 1  
3     i = i - 1;  
4     if i == 8 || i == 7  
5         continue;  
6     end  
7     if i < 3  
8         break;  
9     end  
10    disp(i);  
11 end
```

Outputs:

9 ←

6 ←

5 ←

4 ←

3 ←

M-file

Source code of MATLAB code is stored in .m files. It is essential text file, which means actually you can modify it as for .txt file. There are two kinds of .m file: script and function.

script

Scripts are essentially putting the code line by line to the command window. A script has no specific input and output. It operates on the value of workspace. Note that, if you do not clear the workspace before running a script, all variables in the workspace will be used by script, so of which are not intended, and may cause what you call “XUANXUE” (dark magic).⁵

function

Functions take specific input and produce output. A function only working on its own variables, which you can take it as a separate box. It promotes **code reuse** and **decoupling**.

M-file

Aside: Code Reuse & Decoupling⁶

Code reuse not only means less code you need to produce; more importantly, it means less code you need to take care (debug, modify, or performance improvement). Decoupling means we would like the code be separated into modules, and limit the coupling only to the interfaces, i.e. where modules connect.

Low coupling:



High coupling:



⁶Reference and thanks: some of content is from
<https://github.com/tripack45/VE280-Notes>.

Input & Output

Input