

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.
- Every **warning** (mostly in yellow) means something is potentially wrong. Make sure you understand why it is warning you.
- Please be aware that a program which is right sometimes and wrong the other time, is **wrong**.
- Learn to use Search Engine (e.g. Google, Bing). (fishing \geq fish)

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 every time 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 experience (\Leftrightarrow 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

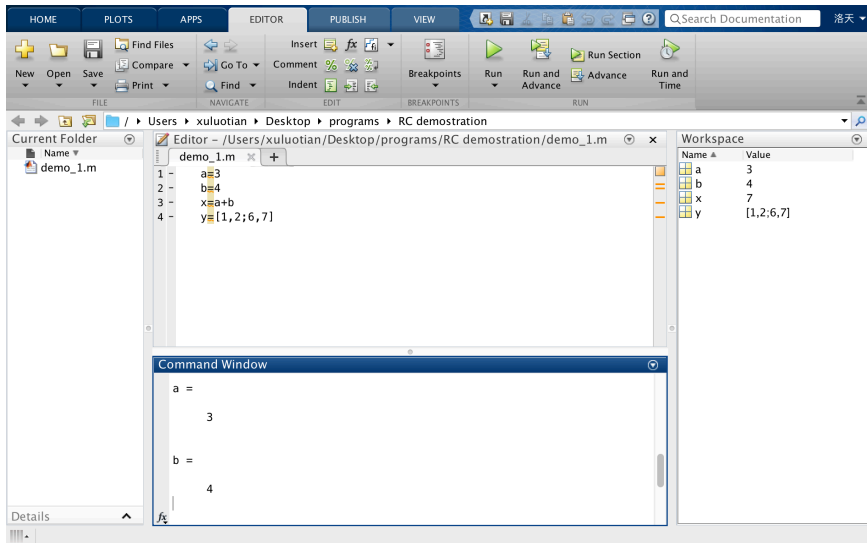
Outline

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

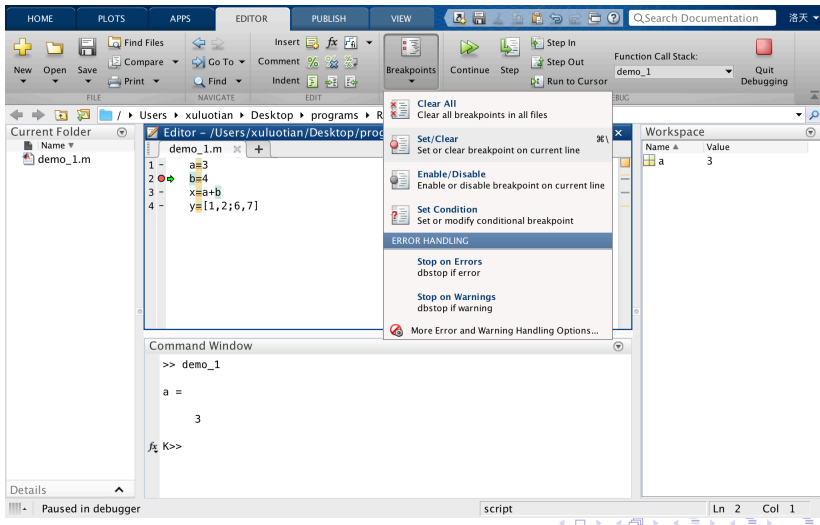
Current Folder (& Environment)

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.

Interface

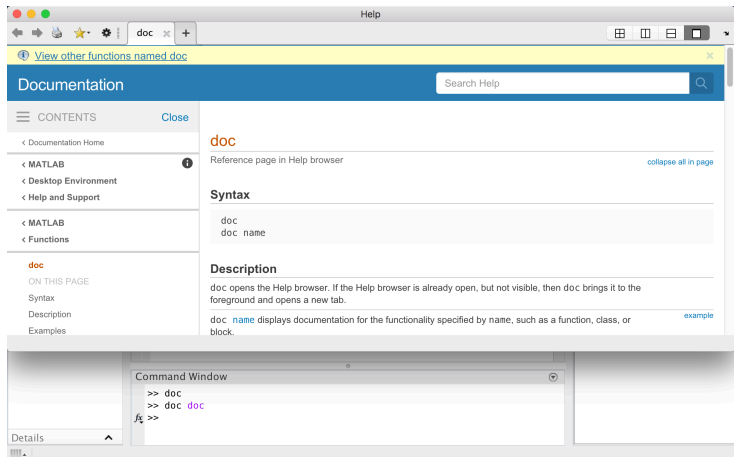
Breakpoint: Useful tool for debugging.



Interface

Documentation: Instructions for using MATLAB

- usage: `doc / doc name` (name = function you want to look up)
- useful in the exam



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. Make your code more readable.
- “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”, “aa”, “aaaa”, “b”, “c”, “d”
 - good variable name: “counter”, “index”, “flag”

Iteration variables “i”, “j”, “k”

- Short, everyone use them
- In MATLAB, sometimes may use “ii”, “jj” instead since “i” and “j” also represent virtual number

¹Not required by VG101 but a really good habit.

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.**

Data Types

Variables has data types. Data type defines how your variable will store in computer. Basic data type you will use:

- Integer (int8, int16, int32, int64)
- Unsigned Integer (uint8, uint16, uint32, uint64)
- Floating-point number (single, double)
- Logical
- Character

Default data type for any number variable is double.

Data Types

Data type conversion:

- new variable = data type (old variable)
- for example: `a = int8 (a)`

Note: some information may be lost.

What will happen if we convert number to character?

- The ASCII code will be used!
- The ASCII code build a bridge between character and integer.

Example:

- » `'a' - 0`
- » `'z' - 'a'`
- » `char('a' + 10)`

ASCII Code

Decimal - Binary - Octal - Hex - ASCII Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOI	36	00100100	044	24	\$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

Matrix

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 matrix?

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

MATLAB allows dynamic changes of array sizes. For example:

- $A = [A; A]$
- $A = [1]; A = [0, A] + [A, 0]$ (What can this do?)
- $A = [1: n];$
for $i = 2 : n$
 $A = [A, [i: n, 1: i - 1]];$
end

Matrix

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 matrix?

- $A(2)$ = The second element in A
If A is a 2-D matrix, it means the second element when arranging element of A by column

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
- $A(:, [i, j]) = A(:, [j, i])$

String

String

When deal with words or sentences, MATLAB uses string. String is displayed in single quotation marks. String is not a new data type. It is a 1-D character matrix. Therefore, matrix operations all apply on strings. ²

Functions for string:

- strcmp, strcmp, strfind
- str2num, num2str, str2double


Check MATLAB document for usage of these functions.

²So the easiest way to concatenate two string is by [], e.g. » ['hello', 'world']

Aside: Array & Data Structure³

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.⁴ 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 complicated data structures.

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

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

M-file: Script & Function

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.”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**.

⁵So you see why we mention adding “clear” in the front of your code is a good habit.

M-file: Script & Function

Function syntax

When defining:

- `function [output1, output2 ...] = functionName (input1, input2 ...)`

When calling (using):

- `[output1, output2 ...] = functionName (input1, input2 ...)`

Note when using function:

- You should use same number of variables to accept the function output values, or the output with no variables to accept will be discarded.
- If you don't specify the variable to accept the function output, MATLAB will create variable "ans" automatically, which is equivalent a single variable to accept function output.
- Function name should be the same as the .m file name. MATLAB will search file name to call the function.

M-file: Script & Function

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>.

Control Flow

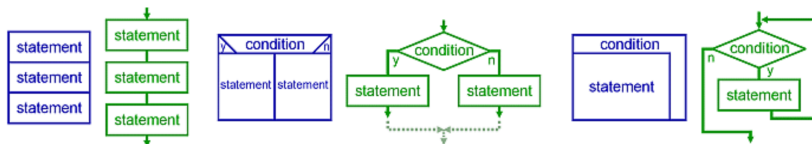
A computer program is like a factory that works on data.

Structured programming

Three basic structures of computer programming

- sequential structure
- selective structure
- cycle structure

Almost all tasks can be done using these three structures.



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 ←

Input & Output

```
A = Input("Please input: ");
```

- Most widely used for command window input.
- Can be used to input anything you want, including matrix.

Other ways of Input:

- `fscanf`: Used for file input, will be discussed later.
- `sscanf`: Used for string input, not used currently.

For more information, you can refer to the help command.

Input & Output

Missing “;” will output the value of current line expression .

- Not recommended unless specified. You will receive deduction if you output redundant messages in homework by not adding “;”.

disp()

- display the value of the variable with a new line
- can display anything you want (number, string, matrix, etc.)
- don't care the output format

fprintf() & sprintf()

- Both make a formatted string
- fprintf: make a formatted string and print it to the file/screen
- sprintf: make a formatted string and store it into a string
- fprintf('hello') \Leftrightarrow disp(sprintf(['hello', '\n']))