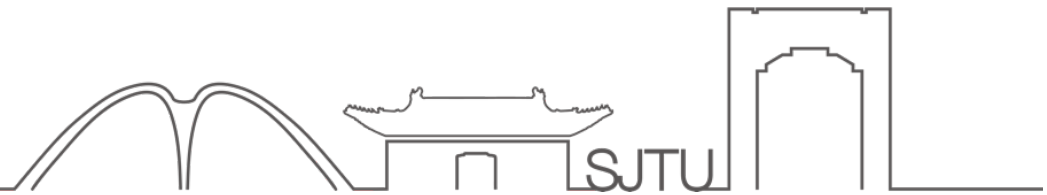
JOINT INSTITUTE
交大密西根学院

VG101: Introduction to Computers and Programming

Lecture 03

Xiaodong Wei



Outline

- Matrices
- M-File
- Debugging
- Data types

Matrices and Arrays

Matrix / Array

- Remember: MATLAB = Matrix Laboratory.
- Array: an ordered series.
- Matrix usually refers to two-dimensional arrays.
- Sometimes we can use either array or matrix.
- Usually it's much faster to use matrix operation other than working with each element sequentially.

8	1	6
3	5	7
4	9	2

Anything special about this matrix?

Entering Matrices/Arrays

- A = [1, 2, 3; 4 5 6; 7, 8, 9]
- B = zeros(3, 2)
- C = ones(4, 3)
- D = [1:100]
- E = [0:0.1:1]

- Access the element in a matrix: A(2,3)

Concatenation

- Concatenation is the process of joining small matrices to make bigger ones.
- In fact, you made your first matrix by concatenating its individual elements.
- The pair of square brackets, [], is the concatenation operator.
- Examples:

```
A = [1, 2, 3];
B = [A;A+5];
```

Practice

- Construct the following matrix

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
4	5	6	7	8
5	6	7	8	9

Deleting Rows and Columns

- You can delete rows and columns from a matrix using just a pair of square brackets.
- Examples

```
A=[1:5];
B=[A; A+1; A+2; A+3; A+4];
B(2,:)=[];
B(:,1)=[];
```

Useful Function: reshape

- Very handy if you want to convert a matrix into a vector, or vice versa.
- Examples

```
A=[1:12];
B=reshape(A, [3,4]);
reshape(B, 1, []);
B(:);
reshape(B, [], 1);
```

sum and *transpose*

- sum: very handy for 1xn or nx1 matrices, i.e., vectors.
- transpose: ’
- Examples

```
D=[1:100];
sum(D);
D’;
A=[1, 2, 3; 4, 5, 6; 7, 8, 9; 10, 11, 12];
A’;
sum(A);
sum(sum(A));
sum(A(:));
sum(A, ’all’);
```

Array Operations

- Operations are done element by element

Operator	Description
+	Addition
-	Subtraction
.*	Element-by-element multiplication
./	Element-by-element division
.\	Element-by-element left division
.^	Element-by-element power
.'	Nonconjugated array transpose

- Examples

  A+5; A-1; A.*2; A./2; A.\2; A.^2; A.'; A';

Matrix Operations

Operator	Description
’	Complex conjugate transpose
*	Matrix product
det	Determinant
inv	Inverse
eig	Eigenvalues

- Examples: solving a linear equation

$$x + y = 1$$
$$2x + 3y = 2$$

$$\begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

```
A=[1,1;2,3];
b=[1;2];
x=inv(A)*b
x=A\b
```

Matrix and Scalar

- Scalar: a single element
- +, -, *, /, ^
- With scalar expansion, MATLAB assigns a specified scalar to all indices in a range.
- Examples

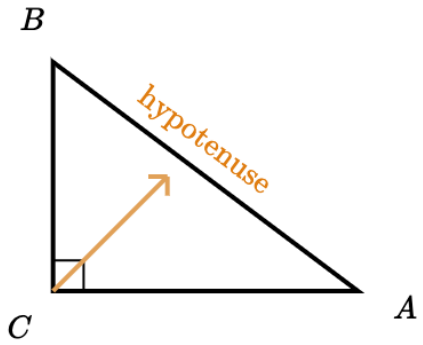
```
A=[1, 2, 3; 4, 5, 6];
A(2,:)=7;
A(2,2:3)=8;
A(2,2:end)=9;
```

M-File

M-Files

- M-File: Program (with .m extension) in MATLAB
  - Contains a series of commands that can be executed together
- You can write MATLAB expressions in an M-File.
- Use semicolon (;) to suppress output.

Example: Calculate the Hypotenuse




```
clear, clc
a = 3;
b = 4;
c_sqr = a^2 + b^2;
c = sqrt(c_sqr)
```

Basic Syntax

- % : comments
- ;
  - Separate rows to create an array: `a=[1;2]`
  - Suppress code output: `a=[1;2]` vs `a=[1;2];`
  - Separate multiple commands on a single line (no results): `a=1; b=2;`
- ,
  - Separate row elements to create an array: `a=[1,2]`
  - Separate subscripts: `A(1,2)`
  - Separate multiple commands on a single line (showing results): `a=1, b=2`
- :
  - Create a vector (1D array): `x=1:10`, `y=1:2:10` (increment of 2)
  - Reshape a matrix into a column vector: `A(:)`
  - Index a range of elements in a particular dimension: `A(2:5,3)`
  - Index all elements in a particular dimension: `A(:,3)`, `A(2,:)`

Entering Long Statement

- If a statement does not fit on one line, use an ellipsis (three dots), ... , followed by Return or Enter to indicate that the statement continues on the next line.
- Example:

```
s = 1 -1/2 + 1/3 -1/4 + 1/5 - 1/6 + 1/7 ...
    - 1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

M-File Side-Effect

- All variables created in a script file are added to the workspace. This may have undesirable effect, because:
  - Variables already existing in the workspace may be overwritten
  - The execution of the script can be affected by the state variables in the workspace.
- Solution: `clear`, `clear all`
  - `clear`: removes all variables from the workspace
  - `clear all`: removes all variables, functions, etc

Input/Output

```
% Example of input/output
% calculate the square of the input number
number = input('Please enter a number:');
number_square = number^2;
fprintf('The square of the number is %f',...
    number_square);
```

RESULT = `input`(PROMPT)

- Displays the `PROMPT` string on the screen, waits for input from the keyboard, evaluates any expressions in the input, and returns the value in `RESULT`

`fprintf`(FORMAT, A, …)

- Formats data of `A` and displays the results on the screen
- `%f`: format specification, floating-point number

Scripts and Functions

- There are two kinds of M-Files:
  - **Scripts**: shown in previous slides, which do not accept input arguments or return output arguments. They operate on data in the workspace.
  - **Functions**: accept input arguments and return output arguments. Internal variables are local to the function.
- Functions can be called in another M-File

Example of a Function

```
function f = myfactorial(n)
% Returns the factorial of n
f = prod(1:n);
end
```

```
>> help myfactorial
```

- Create help text by inserting comments at the beginning of your program.
- For a user-defined function, position the help text immediately below the function definition line (the line with the `function` keyword).

Inputs/Outputs of a Function

- `function` [y1,…,yN] = function_name(x1,…,xM)
  - Declares a function named `function_name` that accepts inputs `x1, …, xM` and returns outputs `y1, …, yN`.
  - This declaration statement must be the first executable line of the function.
  - Valid function names begin with an alphabetic character, and can contain letters, numbers, or underscores.
  - You can save your function:
    - In a function file which contains only function definitions. The name of file must match the name of the first function in the file, e.g., "function_name.m"
    - 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.
  - For readability, use the `end` keyword to indicate the end of each function
- Examples:
  - `function C = FtoC(F)`: 1 input and 1 output
  - `function area = TrArea(a, b, h)`: 3 inputs and 1 output
  - `function [h, d]= motion(v, angle)`: 2 inputs and 2 outputs

Practice

- Implement the function to convert Fahrenheit to Celsius

```
function C = FtoC(F)
```

- Celsius degree = (Fahrenheit degree – 32) * 5/9

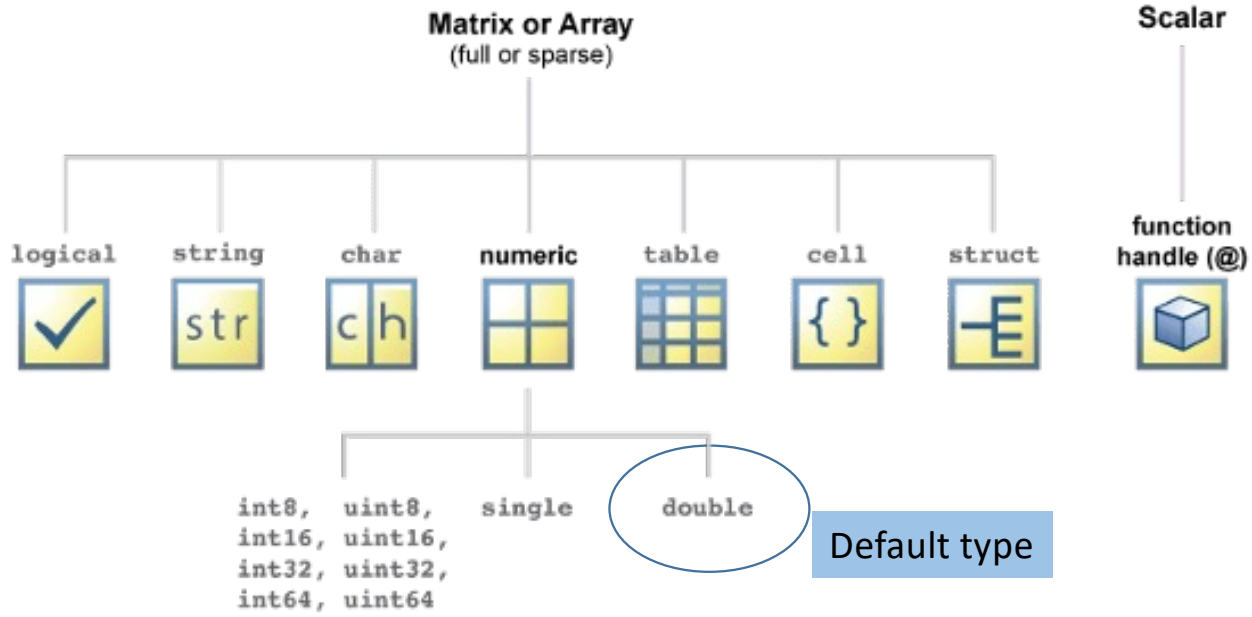
Debugging the M-File

- Insert break points
  - Click on the line number in the editor, highlighted in red
- Run the program step by step
- Continue to next break point
- Example…

Data Types in MATLAB

Data Types

- 17 fundamental data types (`whos` or `class`), in the form of a matrix or array, which can be multidimensional.



Integer

Data Type	Range of Values	Conversion Function
Signed 8-bit integer	-2^7 to 2^7-1	int8
Signed 16-bit integer	-2^{15} to $2^{15}-1$	int16
Signed 32-bit integer	-2^{31} to $2^{31}-1$	int32
Signed 64-bit integer	-2^{63} to $2^{63}-1$	int64
Unsigned 8-bit integer	0 to 2^8-1	uint8
Unsigned 16-bit integer	0 to $2^{16}-1$	uint16
Unsigned 32-bit integer	0 to $2^{32}-1$	uint32
Unsigned 64-bit integer	0 to $2^{64}-1$	uint64

how to convert double to uint8?

Floating-Point Numbers

- Double-precision floating point:
  - 64 bits (8 bytes)
  - maximum and minimum positive values: `realmax`, `realmin`
- Single-precision floating point:
  - 32 bits (4 bytes)
  - maximum and minimum positive values:
    `realmax('single'),realmin('single').`