Introduction to MATLAB Programming with Applications

Zheng-Liang Lu

Department of Computer Science and Information Engineering
National Taiwan University

MATLAB 334
Summer 2020

```
>> Lecture 0
>>
>>              -- Introduction
>>
```

Class Information

- Instructor: 盧政良 (Zheng-Liang Lu)
- Email: arthurzllu@gmail.com
- The course website is

  http://www.csie.ntu.edu.tw/~d00922011/matlab.html.
- All lecture slides are organized in English and will be modified if necessary.

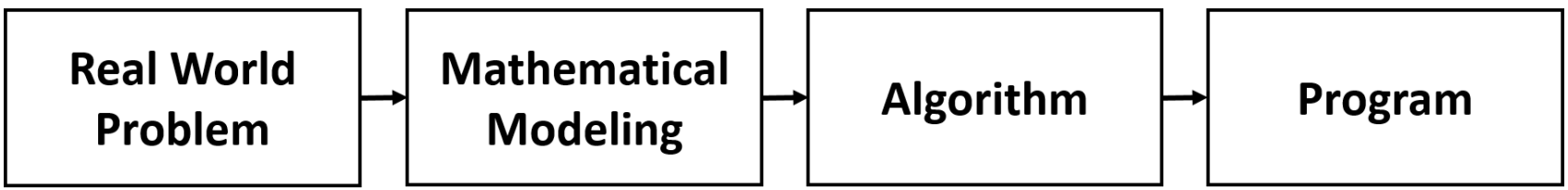
Teaching Philosophy

- I try to lower the barriers to entry.
- I provide resources as many as possible.
- I answer your questions.

Roll Call

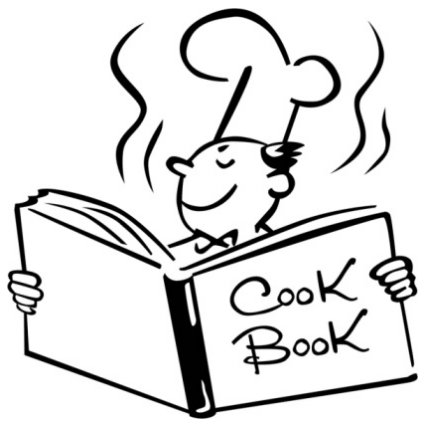
What Can A Program Do?

- A program is an implementation of an algorithm expressed in a specific programming language.



Algorithms In A Nutshell[1]

- An algorithm is a well-defined computational procedure that takes necessary information as input and produces an correct answer as output.
- Simply put, an algorithm is a procedure that solves a specific class of problems, like a recipe or a cookbook.



[1] Also see `http://ed.ted.com/lessons/your-brain-can-solve-algorithms-david-j-malan`.

- An algorithm has properties as follows:
  - Definiteness: all steps are precisely defined.
  - Finiteness: for any input, the algorithm must terminate after a finite number of steps (time).
  - Effectiveness: operations are basic enough (e.g. $+ - \times \div$) to be able to done exactly and in a finite number of steps.
- Note that an algorithm could be expressed not only in programming languages, but also in human languages, flow charts, and pseudo codes.

Example: Greatest Number

- Let A be a list of numbers.
- For example, consider $A = \{1, 7, 9, -2, 4\}$.
- Then it is clear that the answer is 9.
- Now propose an algorithm which finds the greatest element in for any list of numbers.

Input: A.
Output: the greatest element in A.

- Try a top-down approach in your native language?

Optimal Solution

- Let $A(1)$ be the first element of A and so on.
- The symbol $\leftarrow$ is a copy operator from right to left.

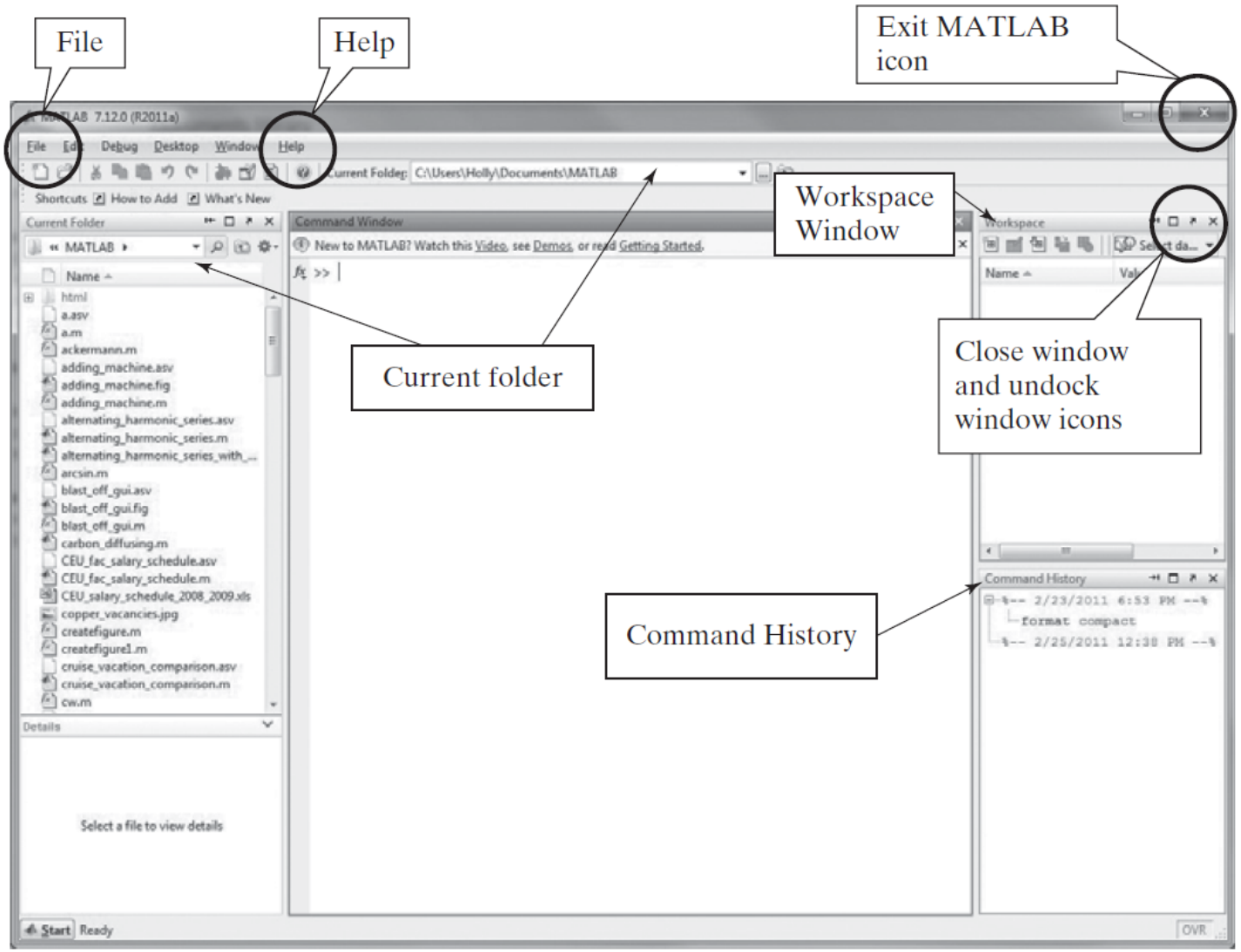
```
1  max <- A(1)
2  for i <- 2 ~ n
3      if A(i) > max
4          max <- A(i)
5      end
6  end
7  return max
```

- In Line 1, why not $\boxed{\text{max} \leftarrow 0}$ but $\boxed{\text{max} \leftarrow A(1)}$?
- You may extend this solution to more questions:
  - Smallest element?
  - Location of the greatest element?

Remarks

- Program design $\approx$ data structures + algorithms
  - Data structures: organize your data in an efficient way
  - Algorithms: process your data so that you can derive the solution
- In some sense, we can say that programming languages are less important than the two above.
- Here we will learn programming concepts and classical algorithms in MATLAB.

MATLAB: An Overview



Command Window

- Let's try a greeting, "Hello, MATLAB."

```
1 >> disp("Hello, MATLAB.");
```

  - **disp** takes the string as input and outputs it on the screen.
  - A string is double-quoted in MATLAB.
- The convention in the slides is as follows:
  - Boxes show the listings for sample programs.
  - Important words and sentences are highlighted in red.
  - Words in blue are reserved words.
  - Bold words in black are functions.

Errors

- MATLAB interrupts your program if an error occurs.
- Don't be frustrated by these red lines.
- Most of these errors detected by MATLAB are syntax errors and runtime errors, which can be avoided by more practices.
- Logic errors cannot be found by the machine itself!

"Why do we fall sir? So that we can learn to pick ourselves up."

– Alfred Pennyworth, *Batman Begins* (2005)

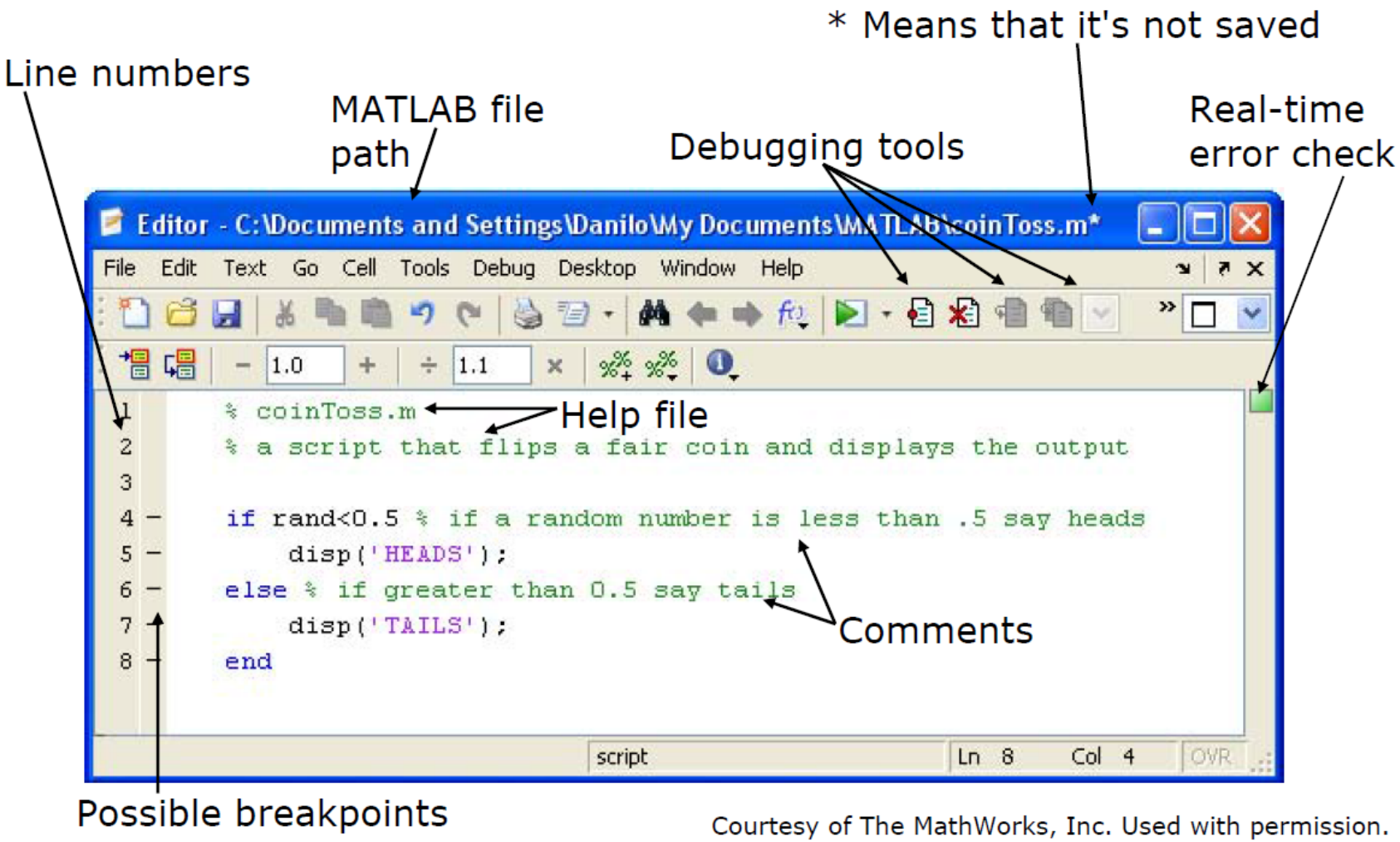
Help Yourself

- **help** followed by the command name shows the usage of that command, for example,

```
1 >> help disp
2  ...
```

- The reference page also provides the detail of commands with fruitful examples.
- Google is your best friend (when you learn by yourself).

Scripts: the Editor



Courtesy of The MathWorks, Inc. Used with permission.

Debut: Your First MATLAB Program

```
1 % This is my first MATLAB program.
2
3 clear; % Clear all variables stored in the workspace.
4 clc; % Clear the screen.
5 % Main program
6 disp("Hello, Matlab.");
```

- The lines which begin with % are treated as comments which won't be executed.
- The command **clear** is used to release all the variables in the workspace[2].
- Use **clc** to clean the command window.

[2]You may delete the variable x by calling **clear** x.

- To run this program, we need to save to a file, for example, helloworld.m.
- Click the *Run* button.[3]
- Alternatively, press F5 for saving the file and executing the program.

```
1 >> helloworld
2
3 Hello, world.
```

[3] If this script is not at the current folder when you execute it, MATLAB will offer two options: (1) change folder, or (2) add to path pool. We often choose (1).

Block Comments

- Press ctrl + r to comment lines.
- Press ctrl + t to de-comment lines. (Useful!)
- The contiguous comment lines starting from the top of file are regarded as the program document.

```
1 >> help helloworld
2
3  This is my first MATLAB program.
```

- We can easily organize the program by these two hot keys during the trial-and-error stage.

```
1 >> Lecture 1
2 >>
3 >>       -- Data, Data Types, and Operators
4 >>
```

Data Types

- Everything is encoded in binary, aka bit (either 0 or 1).
  - Note that 1 byte is equal to 8 bits.
- How many bits do you need to store a value?
  - Not depending on the value itself!
  - The type determine its size![4]
- We have two different numeric types: integers and real numbers (more precisely, floating points[5]).
  - An int value takes 32 bits.
  - A double value takes 64 bits.

[4] Use **class**(x) to check the data type of x.

[5] See IEEE 754 Standard (1985).

Computational Limits

realmax	Returns the largest possible floating-point number used in MATLAB®.	**realmax** **ans =** **1.7977e+308**
realmin	Returns the smallest possible floating-point number used in MATLAB®.	**realmin** **ans =** **2.2251e-308**
intmax	Returns the largest possible integer number used in MATLAB®.	**intmax** **ans =** **2147483647**
intmin	Returns the smallest possible integer number used in MATLAB®.	**intmin** **ans =** **-2147483648**

- It is a common convention to use e to identify a power of 10, e.g. $1e5 = 100,000$ and $-1.2e-5 = -0.000012$.

Numerical Error[7]

- By default, MATLAB treats every numeric value as double.
- However, these double values only approximate real numbers.
  - For example, $0.5 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1 =$? (Why?)
  - Is $3.14 + 1e20 - 1e20$ equal to $3.14 + (1e20 - 1e20)$?
- Finite precision of computations involving floating-points is the main cause of numerical error.[6]

[6] Read `https://news.cnyes.com/news/id/3680649`.

[7] See `https://www.csie.ntu.edu.tw/~cjlin/courses/nm2016/part1.pdf` by Prof. C.-J. Lin.

Variables & Assignments

- A variable is used to keep a value or values.
- A common statement looks like

$$\text{var} = \text{expression},$$

  where var should be a variable and the expression could be a mathematical expression or a function call.[8]
  - For example, suppose $x = 0$.
  - Then run $y = x + 1$ and so $y = 1$. (Trivial.)
- How about $x = x + 1$?
  - The assignment operator $(=)$ is defined to assign a value to the variable, from right to left.
  - This expression acts like a counter, widely used in loops.

[8] To the best of my knowledge, $1 = x$ is not allowed because the left-hand side of the assignment should be a space.

Semicolon

```
1 >> a = 10;
2 >> b = 20;
3 >> a + b
4
5  ans =
6
7       30
```

- The variable *ans* is used by default if you don't assign one for the immediate result.
- If you drop the semicolon (;) in Line 2, then the immediate result will appear in the command window.[9]

[9] Note that you may not drop semicolons in other programming languages. For example, the semicolon is used as the end symbol of a statement in C, C++, and Java.

Variable Naming

- Variable names should always be mnemonic.
- The name length is limited to 63 characters.[10]
- MATLAB is case-sensitive (e.g. A and a are distinct).
- We may use underscores (_) or use CamelCase.[11]
- We avoid to use names of reserved words[12] and built-in functions.
  - $i = \sqrt{-1}$ and $j = \sqrt{-1}$ by default.
  - If you set $i = 1$, then $i = 1$ until you clear i.
  - pi and sin are also the names which are not to be overloaded.

[10] The function **namelengthmax** returns the maximum length allowed in MATLAB.

[11] See `https://en.wikipedia.org/wiki/Camel_case`.

[12] Try **iskeyword**.

Scalar Variables

- The variable x is said to be a scalar variable if $x \in \mathbb{R}^1$ or $\mathbb{C}^1$.
  - The complex field, denoted by $\mathbb{C}$, contains all the complex numbers
    $$a + bi,$$
    where $i = \sqrt{-1}$, and $a, b \in \mathbb{R}$.[13]
  - Try $x = 1 + i$.
- Scalar variables are usually used as model parameters.
- For example,
  $$pi = 3.141592653589793.$$

[13] In math, $\mathbb{C}^1$ is equivalent to $\mathbb{R}^2$. So a complex number actually contains two double values. However, MATLAB treats a complex number as an entity.

Scalar Arithmetic Operators

Operation	Algebraic Syntax	MATLAB® Syntax
Addition	$a + b$	**a + b**
Subtraction	$a - b$	**a – b**
Multiplication	$a \times b$	**a * b**
Division	$\frac{a}{b}$ or $a \div b$	**a / b**
Exponentiation	a^b	**a^b**

Arrays

- An array is a collection of elements, each identified by indices.
- For math, arrays could be
  - row vectors: $u \in \mathbb{R}^{1\times n}$ for $n \in \mathbb{N}$.
  - column vectors: $u \in \mathbb{R}^{n\times 1}$ for $n \in \mathbb{N}$.
  - matrices: $A \in \mathbb{R}^{n\times m}$ for $n, m \in \mathbb{N}$.
- Arrays can be said the simplest (and the most important) one of data structures.

Example

```
>> row_vector = [1, 2, 3] % [] denotes an array.

row_vector =

     1   2   3

>> col_vector = [1; 2; 3] % Semicolons change rows.

col_vector =

     1
     2
     3
```

```
18 >> A = [1 2 3; 4 5 6; 7 8 9]
19
20 A =
21
22      1    2    3
23      4    5    6
24      7    8    9
```

- You can create an $n \times m \times k \times \cdots$ matrix for $n, m, k, \ldots \in \mathbb{N}$.
  - For example, a $10^4 \times 10^4$ matrix takes 762.94 MBytes in memory.[14]
- The number of dimensions of arrays is unlimited without regarding to the limit of the memory space.

[14] More explicit, $10^8 \times 8 \,/\, 2^{20}\mathrm{B} = 762.94\mathrm{MB}$.

Alternatives for Vector Creation

- The function **linspace**(start, end, nPts) generates a vector of uniformly incremented values.[15]

```
>> x = linspace(0, 10, 5)

x =

      0   2.5   5   7.5   10
```

[15] Also try **logspace**.

- You can also create a vector by using the colon operator as follows:

start : step size : end

```
>> x = 0 : 2 : 10

x =

     0   2   4   6   8   10
```

- This is widely used to create index arrays to manipulate arrays.

```
>> x = 1 : 5 % Default step size = 1

x =

     1   2   3   4   5
```

- You could create a null vector, say:

```
>> x = 1 : -1 : 5

x =

     Empty matrix: 1-by-0
```

- It will be useful later!

- Some special matrices can be initialized by the following functions:

zeros(m)	Creates an $m \times m$ matrix of zeros.	**zeros(3)** **ans =** **0 0 0** **0 0 0** **0 0 0**
zeros(m,n)	Creates an $m \times n$ matrix of zeros.	**zeros(2,3)** **ans =** **0 0 0** **0 0 0**
ones(m)	Creates an $m \times m$ matrix of ones.	**ones(3)** **ans =** **1 1 1** **1 1 1** **1 1 1**
ones(m,n)	Creates an $m \times n$ matrix of ones.	**ones(2,3)** **ans =** **1 1 1** **1 1 1**

Array Addressing

- We often use subscripted indexing like the way in math.
- Rarely use linear indexing: column major order.[16]

```
1  >> A(2, 3) % Using subscripted indexing.
2
3  ans =
4
5           6
6
7  >> A(8) % Using linear indexing.
8
9  ans =
10
11          6
```

[16] How could a 1d memory store a 2d array?

```
>> A([1 3], [1 3]) % Range selection.

ans =

     1     3
     7     9

>> A(2, 2 :  end - 1)

ans =

     5
```

```
>> A(:, 2) = [] % Deleting the 2nd column.

A =

     1     3
     4     6
     7     9

>> B = [A, A] % Merge A and A into B.

B =

     1     3     1     3
     4     6     4     6
     7     9     7     9
```

Cells and Cell Arrays[17]

- An array cannot contain data of different types correctly!
- A cell array is a data type with indexed data containers called cells, and each cell can contain any type of data.
- Cell arrays commonly contain either lists of text strings, combinations of text and numbers, or numeric arrays of different sizes.

[17] Basically, cell arrays do not require completely contiguous memory because MATLAB does not allocate any memory for the contents of each cell when the cell array is created. However, each cell requires contiguous memory, as does the cell array header that MATLAB creates to describe the array.

Example[18]

```
>> stock = {"TSMC", 100;
            "GOOG", 200;
            "AAPL", 300}

stock =

3x2 cell array

    {"TSMC"}    {[100]}
    {"GOOG"}    {[200]}
    {"AAPL"}    {[300]}
```

- However, the arithmetic operators cannot be applied to the cell arrays!

[18] Thanks to a lively class discussion (MATLAB263) on March 3, 2016.

Referring to Cell Arrays

- Using curly braces, $\{\cdot\}$ will reference the contents of a cell.

```
1 >> A{3, 1}
2
3  ans =
4
5      AAPL
```

- More details can be found in here.

Load Spreadsheet

- The command **xlsread**(*filename*) reads excel files, for example,

```
1 >> [mat, txt, raw] = xlsread("2330.xlsx")
2 >> [~, ~, raw] = xlsread("2330.xlsx")
```

- By default, it returns a numeric matrix.
- The text part is the 2nd output, separated from the numeric part.
- You may consider the whole spreadsheet by using the 3rd output (stored in a cell array).
- Note that you can use $\sim$ to drop the output value.

Summary[19,20]

- Square brackets $[\cdot]$: only used in array operations.
  - e.g. $numbers = [1 \quad 2 \quad 3 \quad 4]$.
- Curly brackets $\{\cdot\}$: only used in cells.
  - e.g. $A = \{\text{'MATLAB'}, numbers\}$.
- Parentheses $(\cdot)$.
  - Arithmetic, e.g. $(x + y)/z$.
  - Input arguments of functions, e.g. sin(pi / 2), exp(1).
  - Array addressing, e.g. $A(1)$ refers to the first element in array A.

[19] Thanks to a lively class discussion (MATLAB237) on April 16, 2014.

[20] You can refer to this page for more details of special characters.

More Data Structures

- See `https://www.mathworks.com/help/matlab/data-types_data-types.html`.[21]

[21] Some of them may not be available if your version is out-of-date.

Vectorization[22]

- MATLAB favors array operations.
- When two arrays have the same dimensions, addition, subtraction, multiplication, and division apply on an element-by-element basis.
- For example,

```
>> x = [1, 2, 3];
>> y = [4, 5, 6];
>> x + y

ans =

     5     7     9
```

[22] More about vectorization.

Element-By-Element Operations

Symbol	Operation	Form	Example
+	Scalar-array addition	A + b	[6,3]+2=[8,5]
−	Scalar-array subtraction	A − b	[8,3]−5=[3,−2]
+	Array addition	A + B	[6,5]+[4,8]=[10,13]
−	Array subtraction	A − B	[6,5]−[4,8]=[2,−3]
.*	Array multiplication	A.*B	[3,5].*[4,8]=[12,40]
./	Array right division	A./B	[2,5]./[4,8]=[2/4,5/8]
.\	Array left division	A.\B	[2,5].\[4,8]=[2\4,5\8]
.^	Array exponentiation	A.^B	[3,5].^2=[3^2,5^2] 2.^[3,5]=[2^3,2^5] [3,5].^[2,4]=[3^2,5^4]

- The left division is used in the inverse matrix problems.[23]

[23] We will visit this in the chapter of matrix computation.

Relational Operators[24]

Relational Operator	Interpretation
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
~=	not equal to

- Note that relational operators make comparisons between two arrays of equal size.

[24] See Table 8.1 in Moore, p. 274.

Logical Values

- For example,

```
>> x = 1; y = 2;
>> x == y

ans =

     0
```

- In general, the numeric number 0 is regarded as false while 1 (even any nonzero number) is regarded as true.
- The function **true** and **false** represent logical true and false, respectively.[25]

[25] The usage of **true** and **false** is similar to **zeros**.

Filtering

- Logical arrays are often used as masks (or filters) to manipulate arrays.

```
>> scores = { "Arthur", 50;
                   "Bob", 60;
                 "Cynthia", 70};
>> mask = [scores{:, 2}] >= 60

mask =

     0   1   1

>> scores(mask, 1)

ans =

     {"Bob"     }
     {"Cynthia"}
```

Logical Operators

- Assume $x = 0$.
- How about $1 < x < 3$? (Surprising!)

Logical operator	Name	Description
& Example: $A\&B$	AND	Operates on two operands (A and B). If both are true, the result is true (1), otherwise the result is false (0).
| Example: $A\|B$	OR	Operates on two operands (A and B). If either one, or both are true, the result is true (1), otherwise (both are false) the result is false (0).
~ Example: $\sim A$	NOT	Operates on one operand (A). Gives the opposite of the operand. True (1) if the operand is false, and false (0) if the operand is true.

Truth Table[26]

- Let A and B be two logical variables.
- Then you can find the truth table for logical operators as follows:

A	B	∼A	A&B	A | B
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

[26] Note that the basic instructions, such as the plus operator, are implemented by logic gates. See any textbook for digital circuit design.

Exercise: & vs. ==[27]

```
>> u = [0 2 0 4];
>> v = [0 0 3 4];
>> u == v

ans =

     1     0     0     1

>> u & v

ans =

     0     0     0     1
```

[27] Thanks to a lively class discussion (MATLAB-237) on April 16, 2014.

Precedence of Operators[28]

Operators	Precedence
parentheses: ( )	Highest
transpose and power: ' , ^ , .^	
unary: negation (-), not (~)	
multiplication, division *, /, \, .*, ./, .\	
addition, subtraction +, -	
relational <, <=, >, >=, ==, ~=	
element-wise and &	
element-wise or |	
and && (scalars)	
or || (scalars)	
assignment =	Lowest

[28] See Table 1.2 Operator Precedence Rules in Attaway, p. 25.