```
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![1]
- We have two different numeric types: integers and real numbers (more precisely, floating points[2]).
  - An int value takes 32 bits.
  - A double value takes 64 bits.

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

[2] 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[4]

- 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.[3]

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

[4] 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.[5]
  - 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.

[5] 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.[6]

[6] 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.[7]
- MATLAB is case-sensitive (e.g. A and a are distinct).
- We may use underscores (_) or use CamelCase.[8]
- We avoid to use names of reserved words[9] 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.

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

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

[9] 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 $a, b \in \mathbb{R}$.[10]
  - Try $x = 1 + i$.
- Scalar variables are usually used as model parameters.
- For example,
  $$pi = 3.141592653589793.$$

[10] 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.[11]
- The number of dimensions of arrays is unlimited without regarding to the limit of the memory space.

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

Alternatives for Vector Creation

- The function **linspace**(start, end, nPts) generates a vector of uniformly incremented values.[12]

```
1 >> x = linspace(0, 10, 5)
2
3  x =
4
5       0   2.5   5   7.5   10
```

[12] 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.[13]

```
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
```

[13] 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[14]

- 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.

[14] 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[15]

```
>> 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!

[15] 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[16,17]

- 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.

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

[17] 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`.[18]

[18] Some of them may not be available if your version is out-of-date.