Basics of MATLAB

Jiseok Chae

Department of Mathematical Sciences
KAIST

Week 2

Welcome to the MATLAB session!

Like today, every Friday with a recitation class, a video will be uploaded.

Brief explanations about the functionalities of MATLAB will be discussed in the MATLAB session.

Contents

1 Hello MATLAB!

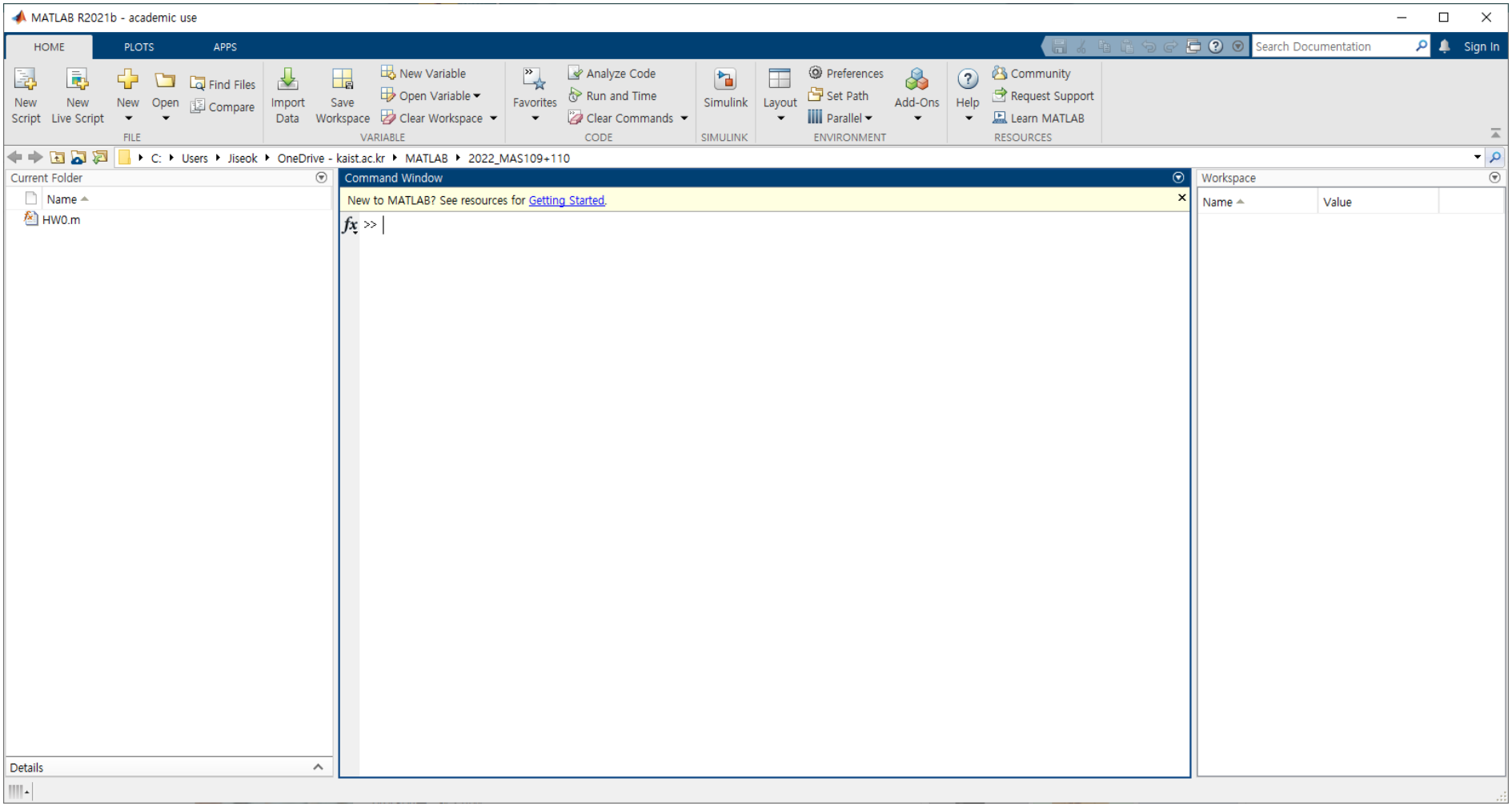
2 Variables

3 Matrices and Vectors

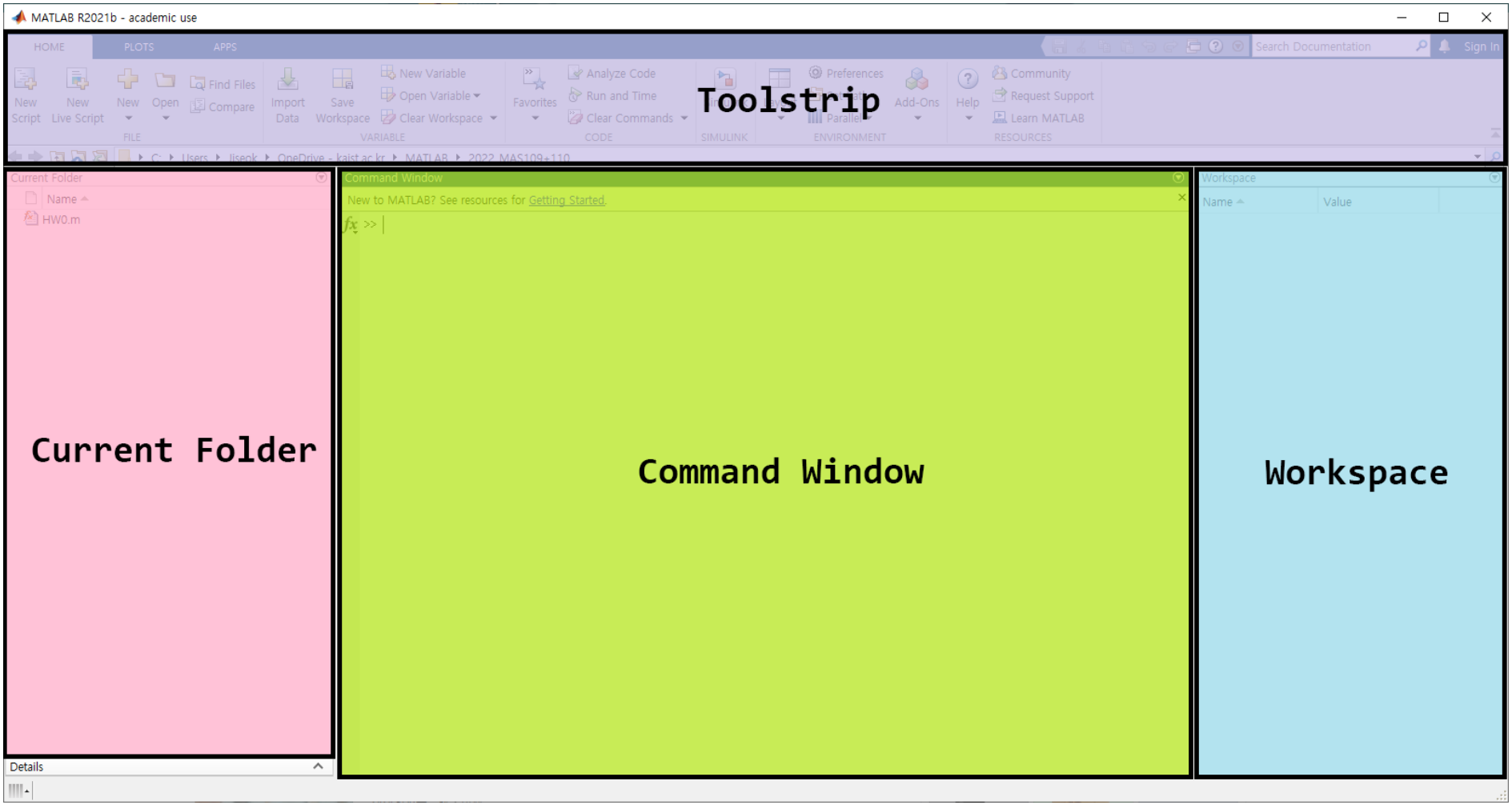
MATLAB is a software package with an interactive environment.

MATLAB is specialized for matrix computations, but also a powerful tool for function and data plotting, data analysis, mathematical modeling, etc.

If you start MATLAB you will see a window that looks something like this:



Each part has the following name:



Commands are expressions you enter so that MATLAB evaluates it to you.

If you first launch MATLAB, the command window will look like:

>>

The symbol >> indicates that the MATLAB is ready to operate.

You can use MATLAB as a highly sophisticated calculator.

Let us type in the following expression...

```
>> 2 + 3
```

...and press the *Enter* key in your keyboard.

You will see the following result.

```
>> 2 + 3

ans =

     5
```

The command window contains both your input and MATLAB's output.

In the slides we will distinguish them by highlighting the inputs by color.

Other basic arithmetic operations can be done similarily.

operation	input	output
Addition	`2 + 3`	5
Subtraction	`9 - 1`	8
Multiplication	`6 * 5`	30
Division	`7 / 4`	1.7500
Exponentiation	`8 ^ 2`	64

There are several predefined functions and constants, not limited to below:

```
>> sin(pi) + sqrt(2)

ans =

    1.4142

>> log(exp(1))

ans =

     1
```

`pi` is the constant π, `sqrt` is the square root, `log` is the natural log, and `exp` is the base-e exponentiation.

You can use the percent symbol (%) to make a comment.

Anything after the symbol % in that line will be ignored by MATLAB.

A comment is colored green in MATLAB by default, so we will also do so.

```
>> 9 * 7 % / 2 ; The division will not be executed!

ans =

     63
```

Contents

1 Hello MATLAB!

2 Variables

3 Matrices and Vectors

Variables are like named boxes that contain values.

To declare a variable with the name `x` which contains the value 5, you simply do the following.

```
>> x = 5

x =

     5
```

Variable names must start with an alphabet, and should be consisted of alphabets, numbers, and underbars.

The simple rule is that MATLAB evaulates the expression on the right hand side, and stores it in the left hand side.

```
>> x = 5

x =

     5

>> x = x + 17

x =

    22
```

So = does not really mean an equality; it is rather a substitution.

A semicolon (;) indicates the end of a command, and suppresses the printing of the output.

You can define variables with values derived from different variables.

```
>> hello_KAIST = 142; welcome = 857;
>> neopjuk = hello_KAIST + welcome

neopjuk =

     999

>> welcome = 1;
>> neopjuk

neopjuk =

     999
```

Actually, `ans` is also a variable, automatically generated by MATLAB.

```
>> 40 + 1

ans   =

      41

>> ans * 271

ans  =

      11111
```

And yes, 41×271 is equal to 11111.

Contents

1 Hello MATLAB!

2 Variables

3 Matrices and Vectors

This is how you declare a matrix in MATLAB:

```
>> A = [1, 9, 7, 1; 2, 0, 2, 4]

A =

     1     9     7     1
     2     0     2     4
```

Now `A` contains the 2×4 matrix $\begin{bmatrix} 1 & 9 & 7 & 1 \\ 2 & 0 & 2 & 4 \end{bmatrix}$.

A comma (`,`) separates entries. A semicolon (`;`) separates rows.

The same can be done as the following, in order to enhance the readability of your input.

```
>> A = [1, 9, 7, 1;
        2, 0, 2, 4]

A =

     1     9     7     1
     2     0     2     4
```

Row vectors are matrices with only 1 row.
Column vectors are matrices with only 1 column.

```
>> a = [2, 5];
>> b = [1; 0; 9];
```

a contains the row vector $\begin{bmatrix} 2 & 5 \end{bmatrix}$. b contains the column vector $\begin{bmatrix} 1 \\ 0 \\ 9 \end{bmatrix}$.

There are functions that help you generate special matrices.

- `zeros(n)` makes an $n \times n$ zero matrix.
- `zeros(n, m)` makes an $n \times m$ zero matrix.
- `ones(n)` makes an $n \times n$ matrix with all entries 1.
- `ones(n, m)` makes an $n \times m$ matrix with all entries 1.
- `eye(n)` makes an $n \times n$ identity matrix.

You can add, subtract, and multiply vectors and matrices with compatible sizes using +, −, and ∗.

```
>> A = [1, -3, 2;
        4, 0, -5];
>> x = [3; 1; 2];
>> A * x

ans =

     4
     2
```

If you add the dot (.) in front of *, MATLAB computes the elementwise product.

```
>> A = [1, 2; 3, 4]; B = [5, 6; 7, 8];
>> A .* B

ans =

     5    12
    21    32
```

A dot (.) in general indicates an elementwise operation.

```
>> A = [1, 2; 3, 4]; B = [5, 6; 7, 8];
>> A ./ B

ans =

    0.2000    0.3333
    0.4286    0.5000

>> A .^ 2

ans =

     1     4
     9    16
```

Some functions, when applied to matrix, is applied elementwise.

```
>> A = [1, 2; 3, 4];
>> sqrt(A)

ans =

    1.0000    1.4142
    1.7321    2.0000
```

Question : what would be the result of `sin(A)`? `exp(A)`?

To get the transpose, use an apostrophe (').

```
>> A = [1, 2; 3, 4];
>> A'

ans =

     1     3
     2     4
```

Let's see more examples of matrix operations. Let `A` be a matrix, then

- `inv(A)` computes the inverse of `A`.
- `A^k` computes the k^{th} power of `A`.
  - `A^0` will be the identity matrix.
  - `A^-1` is the same as `inv(A)`.
- `rref(A)` computes the reduced row echelon form of `A`.

You can access to the entries using parentheses (`()`).

```
>> a = [2, 5]; b = [1; 0; 9];
>> a(1)

ans =

     2

>> b(3)

ans =

     9
```

`a(1)` is the first entry of `a`, hence 2. `b(3)` is the third entry of `b`, hence 9.

You can access to the entries using parentheses (`()`).

```
>> A = [1, 2, 3;
        4, 5, 6;
        7, 8, 9];

>> A(3, 2)

ans =

     8
```

`A(3, 2)` means, select from `A` only the 3^{rd} row and the 2^{nd} column. Therefore, `A(3, 2)` means the $(3, 2)$ entry of `A`, which is $A_{32} = 8$.

Instead of specifying the index, a colon (`:`) can be used to say "select all".

```
>> A = [1, 2, 3;
        4, 5, 6;
        7, 8, 9];

>> A(2, :)

ans =

     4     5     6
```

`A(2, :)` means, select from `A` only the 2^{nd} row and all the columns. Therefore, `A(2, :)` is the 2^{nd} row vector of `A`.

Question: what would be the result of `A(:, 1)`?

More complex selections can be made.

```
>> A = [1, 2, 3;
        4, 5, 6;
        7, 8, 9];

>> A([1, 3], 2)

ans =

     2
     8
```

`A([1, 3], 2)` means, select from `A` only the 1^{st} and 3^{rd} row, and the 2^{nd} column.

A colon (:) can be used to denote a consecutive range of indices.

```
>> A = [1, 2, 3, 4;
        5, 6, 7, 8;
        9, 10, 11, 12];

>> A(3, 2:4)

ans =

      10      11      12
```

This is because a:b is automatically translated to the $(b - a + 1)$-dim'l row vector $[a\ a+1\ \ldots\ b]$.

Thank you!