L02: First Steps in Matlab

Arithmetic and logical expressions; Variables and assignment

Lucas A. J. Bastien

E7 Spring 2017, University of California at Berkeley

January 20, 2017

Version: release

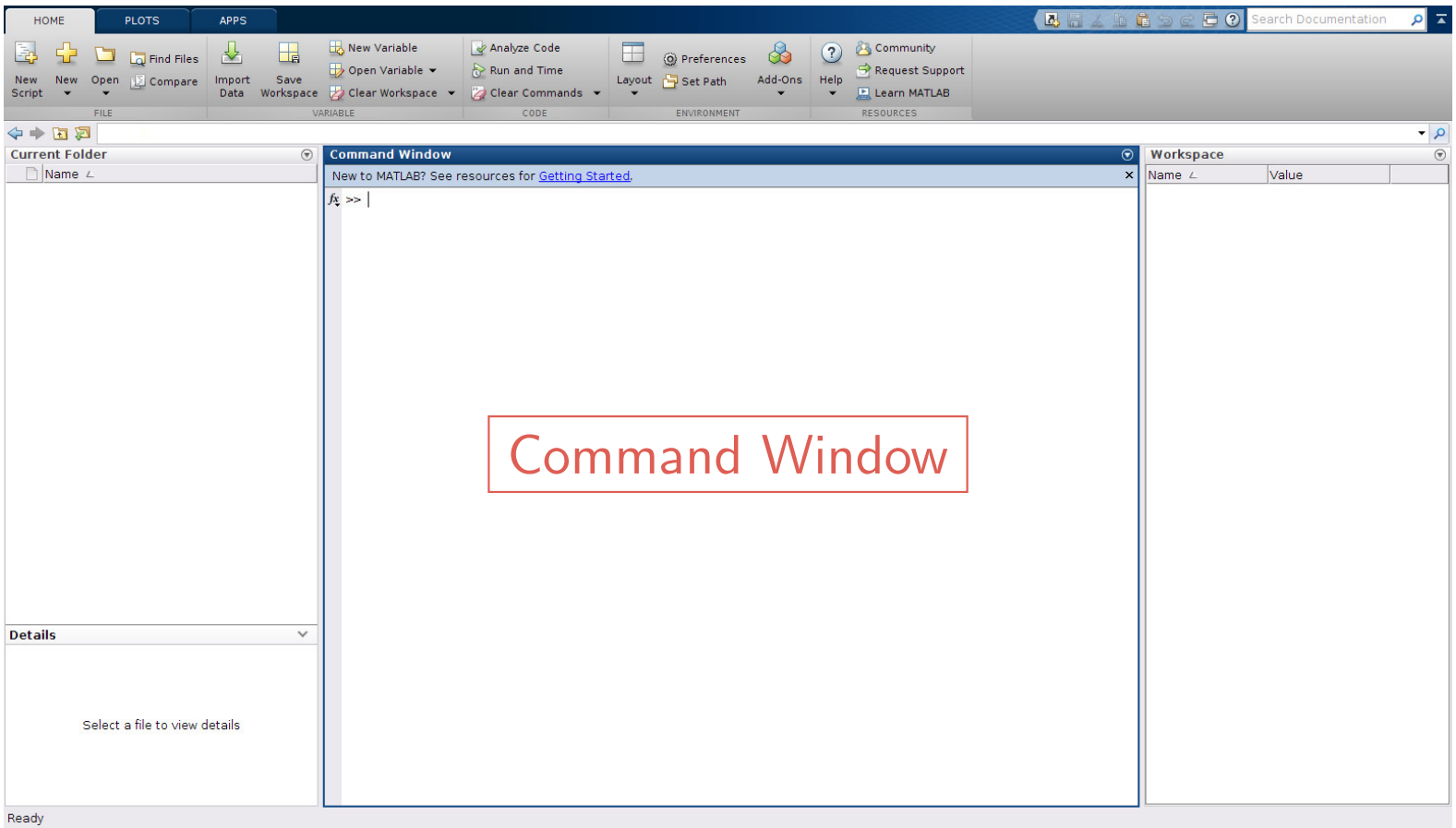
Announcements

- Lab 01 was posted yesterday. It is **due on Friday January 27^{th} at 12 pm**
- Slides from the first two lectures will be posted on bCourses shortly
- Instructions for accessing an electronic version of the textbook have been posted in a bCourses announcement
- Reading from the textbook:
  - For Lab 01: Chapter 1; Chapter 2 (section 2.1 only)
  - For next week: Chapter 2 (sections 2.2 and 2.3 only); Chapter 3

Overview of the Matlab Interface

For now, we focus on the **Command Window**. We will use it to:

- Issue commands for Matlab to execute
- Manipulate data
- Call functions and programs



Matlab as a calculator

Arithmetic: study of numbers and operations on numbers

Some arithmetic operators in Matlab:

- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- ^ (exponentiation)

```
>> 4 + 3
ans =
    7
>> 4 - 3
ans =
    1
```

```
>> 4 * 3
ans =
    12
>> 4 / 3
ans =
    1.3333
>> 4 ^ 3
ans =
    64
```

Order of operations

In what order are operations calculated?

Things in parentheses	↑	evaluated first (higher **precedence**)
Exponentiation (^)		
Multiplication (*); Division (/)		
Addition (+); Subtraction (-)	↓	evaluated last (lower **precedence**)

```
>> 4 + 3 * 5 + 2
ans =
    21
```

```
>> (4 + 3) * (5 + 2)
ans =
    49
```

Operators of equal precedence are evaluated left to right

```
>> 3 * 6 / 2 * 3
ans =
    27
```

```
>> 3 * 6 / (2 * 3)
ans =
    3
```

Use spaces wisely

You can add as many spaces around operators as you want. They do **not** change the order of operations

```
>> 4 + 3                                    * 2
ans =
    10
```

Use spaces wisely, to make your code more readable

- *e.g.*, emphasize order of operations

```
>> 4 + 3*7
ans =
    25
```

is probably more readable than:

```
>> 4+3 * 7
ans =
    25
```

Confusing style?

Which Matlab arithmetic expression yields the value of 2?

(A) `8*3 / 6*2`

(B) `8 ^ 1/3`

(C) `((5^2 – 3^2)^0.5)^0.5`

(D) All of the above

(E) None of the above

Again, use spaces wisely to make your code more readable!

`8*3 / 6*2` → `8 * 3 / 6 * 2` → `8`

`8 ^ 1/3` → `8^1 / 3` → `2.6667`

When things don't work

What is going to happen if I issue the following command?

```
>> 5 + * 3
```

(A) Matlab tells me the result is 8

(B) Matlab tells me the result is 15

(C) Matlab crashes

(D) Matlab nicely explains to me what I am doing wrong

Answer is D, with the following error message:

Unexpected MATLAB operator

→ **Understanding error messages is a very useful skill!**

Built-in functions

Matlab has many built-in functions (see sample below). We will learn more about Matlab functions next week.

```
>> cos(0.5)
ans =
    0.8776
>> exp(0.5)
ans =
    1.6487
```

Name	Mathematical function	Matlab function
cosine	$x \mapsto \cos(x)$	`cos`
arccosine	$x \mapsto \arccos(x)$	`acos`
sine	$x \mapsto \sin(x)$	`sin`
arcsine	$x \mapsto \arcsin(x)$	`asin`
square root	$x \mapsto \sqrt{x}$	`sqrt`
exponential	$x \mapsto e^x$	`exp`
natural log	$x \mapsto \ln(x)$	`log`
log base 10	$x \mapsto \log_{10}(x)$	`log10`

Variables and assignment

What if we want to store information for later use?

- We can use variables. A variable is a place holder for data. Variables have names.

To give a value to a variable, we use the assignment operator $=$

```
>> a = 2;
>> b = 4;
>> a + b
ans =
    6
```

Valid variable name: starts with a letter, contains only letters (upper and/or lower case), digits, and/or underscores.

Some names cannot be used for variable names

- Use command "`iskeyword`" at the command prompt to see the list

Assignment operator

The Matlab assignment operator = is very different from the equal sign in math.

In math, the equal sign is used to state the equality between two numbers.

The Matlab assignment operator does the following (in that order):

1. Calculates the quantity that is located to the right of =
2. Assigns that value to the variable located to the left of =

```
>> a = 5;
>> b = a + 5;
>> a = a + cos(b);
>> a
     a =
         4.1609
```

Assignment operator (continued)

In math, whatever a and b are, $a = b$ and $b = a$ are equivalent.

- *i.e.* $=$ is symmetric.

In Matlab, = is not symmetric!!!

```
>> my_variable = 10;
>> my_variable + 5
ans =
    15

>> 10 = my_variable;
```

Entering the last command above at the Matlab command prompt yields the following error message:

Error: The expression to the left of the equals sign is not a valid target for an assignment

Use sensible variable names

Example with the ideal gas law ($PV = nRT$)

```
>> pressure = 101325;
>> volume = 1;
>> gas_constant = 8.314;
>> temperature = 293;

>> moles = pressure * volume / gas_constant / temperature;
```

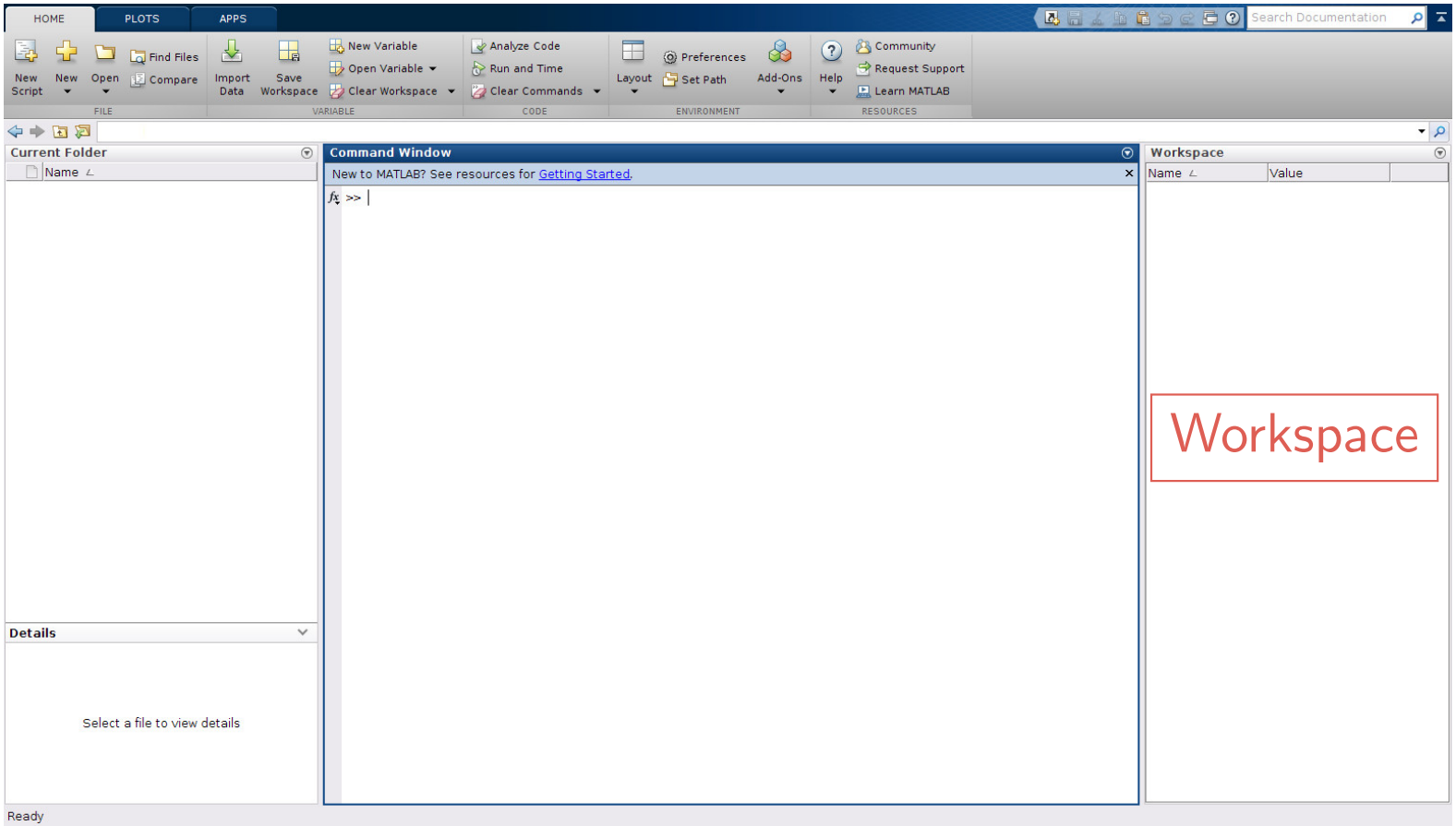
is probably better than:

```
>> a = 101325;
>> lot = 1;
>> of = 8.314;
>> fun = 293;

>> E7 = a * lot / of / fun;
```

The Matlab Workspace

The Workspace is the collection of variables that are already defined.



The Matlab Workspace (continued)

Useful commands:

- **`who`**, **`whos`**: list all the variables currently defined in the Workspace
- **`clear`**: remove selected variables from the Workspace (*e.g.*, clear var1, var2)
- **`clear all`**: remove all variables from the Workspace

What if I use a variable that is not defined?

(A) Matlab defines the variable and gives it the value 0

(B) Matlab defines the variable and gives it the value I was thinking about (it knows!)

(C) Matlab throws an error

(D) Matlab asks me what value it should give to the variable

Answer: C, with the error message: Undefined function or variable

Fundamental data classes

Use the function "class" to inquire the class of specific data

```
>> my_number = 10;
>> my_character_string = 'Hello World!';
>> my_logical_value = true;

>> class(my_number)
ans =
    double

>> class(my_character_string)
ans =
    char

>> class(my_logical_value)
ans =
    logical
```

Logical class and relational operators

In Matlab:

- "true" is represented by `1`
- "false" is represented by `0`

```
>> 3 > 4
ans =
     0
>> 5 == 5
ans =
     1
```

```
>> class(5 == 5)
ans =
    logical
>> class(1)
ans =
    double
```

Relational operators (to compare two quantities):

- > (strictly greater than)
- >= (greater than or equal to)
- == is equal
- < (strictly less than)
- <= (less than or equal to)
- ~= is not equal

Logical operators

& (and)

	false	true
false	false	false
true	false	true

(a & b) is true if and only if both a and b are true

| (inclusive or)

	false	true
false	false	true
true	true	true

(a | b) is true if and only if either a and/or b is true

~ (not)

~a is true if and only if a is false

```
>> (3>2) & (2>1)
ans =
     1

>> (3<2) & (2>1)
ans =
     0
```

```
>> (3>2) | (2>1)
ans =
     1

>> (3<2) | (2>1)
ans =
     1
```

```
>> ~(3<2)
ans =
     1

>> ~((3>2) | (2<1))
ans =
     0
```

Operator precedence

Things in parentheses	↑	higher precedence
Exponentiation (^)		
Logical negation (~)		
Multiplication (*); Division (/)		
Relational operators (< <= > >= == ~=)		
Logical “and” (&)		
Logical “or” (|)	↓	lower precedence

Full list at https://www.mathworks.com/help/matlab/matlab_prog/operator-precedence.html

Logical expressions (practice)

What logical values do the following logical expressions yield?

(A) ~(true & false) ← **true**

(B) ~(true | false) ← **false**

(C) true & false | true & false & true ← **false**

(D) true | (false & (true & false) | false | true) ← **true**

Special quantities

- **`Inf`**: infinity (including when the number is finite but its magnitude too large for Matlab)
- **`NaN`**: not a number

```
>> 1 / 0;
ans =
    Inf
>> -1 / 0;
ans =
    -Inf
>> exp(1000000)
ans =
    Inf
```

```
>> 0 / 0;
ans =
    NaN
>> 1 + NaN
ans =
    NaN
>> NaN == NaN
ans =
    0
```

Today we learned

- Using Matlab as a calculator
- Arithmetic and logical expressions
- How to create and use variables
- How to manage the workspace
- A few different error messages

Feedback?

- Pack first (30 seconds).
- What went well? (1 minute)
- What could be improved? (1 minute)