Matlab Basics

Yaara Erez

MRC Cognition and Brain Sciences Unit

November 2015

MatLab – Matrix Laboratory

- Programming environment based on matrix representations.
- Mainly useful for data analysis, simulations (research, engineering).
- Contains a large set of ready-to-use functions.
- Easy graphics.

Why is programming important?

- It gives you the power to do whatever you want with your data / experiment / simulations, without being limited by off-the-shelf software or scripts written by others.

  → It significantly enhances your capabilities as researchers.

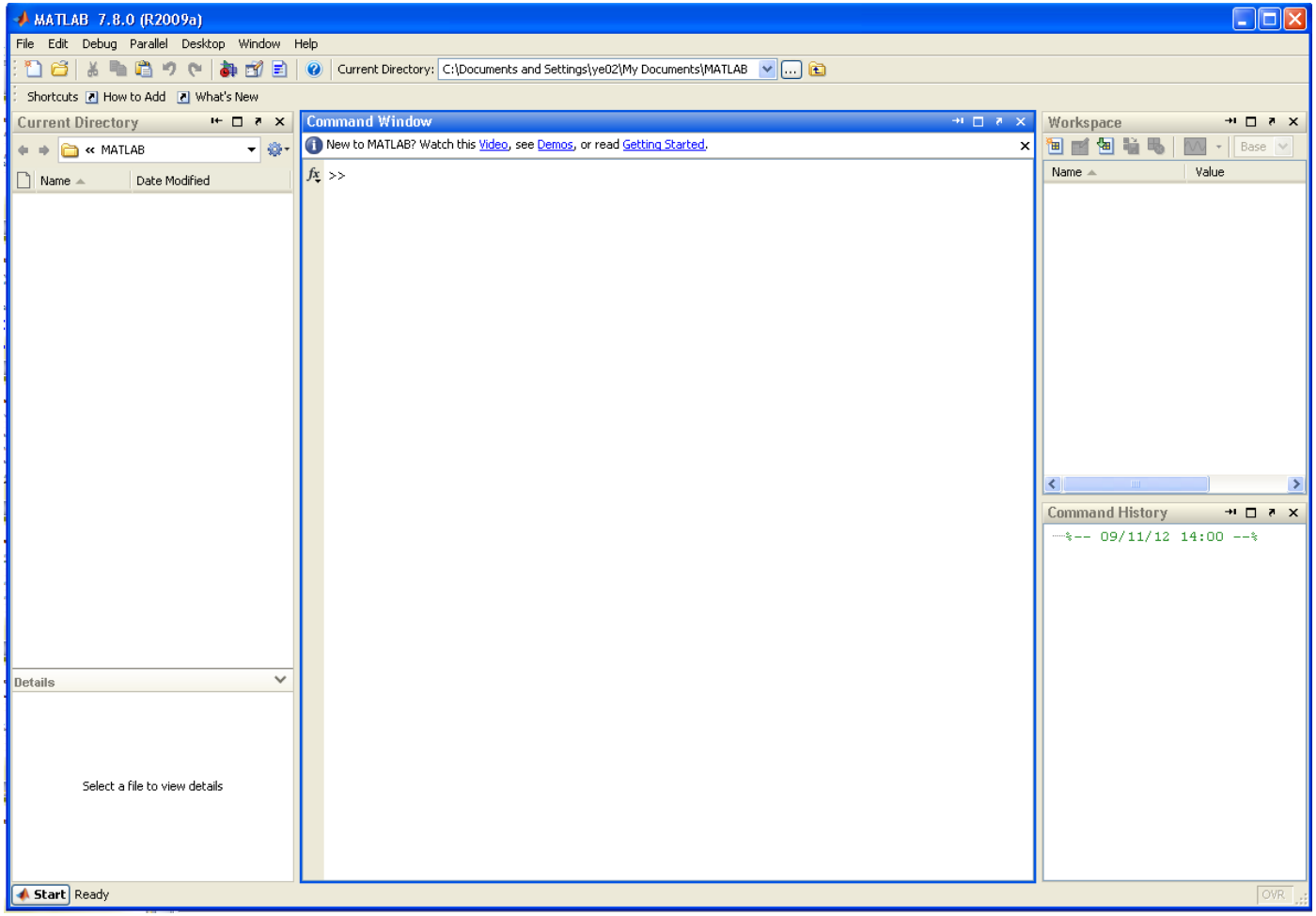
- Some of the commonly used software in research is Matlab-based (like SPM for fMRI/MEG data analysis).

This talk

- Matlab basics
  - Matlab programming environment
  - Variables
  - Editor, scripts, functions

- Practice practice practice!

Matlab programming environment



‘Current Directory’

- The **current directory** is the directory, or path, to which Matlab currently refers when reading/writing files, unless a different path is specified for a file.

- When opening Matlab, it is recommended to change the **current directory** to the one that you are working with.
  - It makes it easier to manage/find/save files.

- Use **full paths** when referring to files whenever possible, to avoid any confusion with files being saved at the current directory.

A few words about **syntax**

- **Syntax** is “the **set of rules** that define the **combinations of symbols** that are considered to be correctly structured programs in a programming language”.

- In other words, it is the **vocabulary and grammar** with which we write our code, such that it will be **unambiguously** understandable by the programming language.

  - When defining a variable, refer to it later *exactly* in the same name. Tip: use **copy & paste**.
  - **Typos** are unacceptable.
  - Matlab is **case-sensitive**.

Variables

- A **variable**: a **place** in memory with a **name** that contains a **value**.

- Variables types – 2 basic types in Matlab (roughly speaking):
  - **Numeric**: single element (scalar), array, multi-dimensional array.
  - **Text**: character, string (array of characters).

Defining numeric variables

- `x = 1;` (scalar, integer)
- `numSubjects = 8;` (meaningful name)
- `myScalar = 1.1;` (scalar, rational (decimal) number)
- `myVec = [1  2  3];` (one-dimensional array)
- `myVec = [1.2  2  3];` (one-dimensional array with mixed integers and rational numbers)

Semicolon (;) at the end of a command prevents echo in the command line

Arrays and indexing

- **Array** – a set of ordered elements.
- **Indexing** – Every element in the array has a place called **index**.
  - The i-th element is the element in the i-th place.
- Defining arrays – by assignment:
  - myVec = [3 1 7 9 4]; → the index of 7 is 3
- **Retrieval** – Getting an element from a specific index in the array.
  - *arrayName(index)*
  - myVec(3) → 7
- **Assignment** – an element can be replaced:
  - *arrayName(index) = newValue*
  - myVec(3) = 5 → myVec = [3 1 5 9 4]

myVec(1)	myVec(2)	myVec(3)	myVec(4)	myVec(5)
3	1	7	9	4

Example

- MatlabBasics.m (examples 1-4)

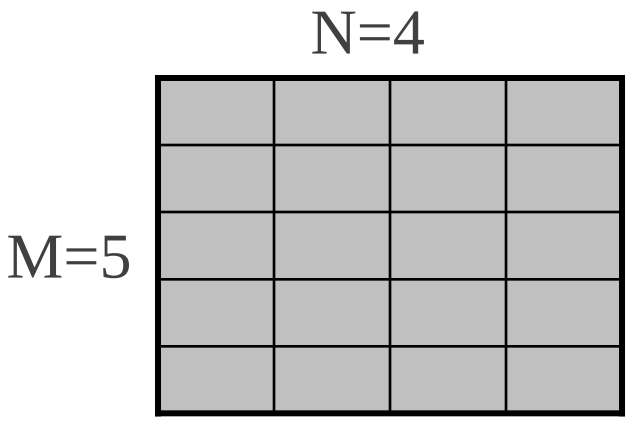
Practice 1

- Open Matlab and change the current directory to a folder of your choice.
- In the command window, do the following:
  - Create a 1x5 array with numeric values as you like.
  - Find the variable in the workspace and double-click it to see its content.
  - Change the value of the 3rd element in the array. Make sure you can see this change in the workspace.
  - Delete the 4th element in the array.
  - Use ‘size’ function to check for the size of the array.
  - Use ‘length’ function to check for the length of the array.
  - Clear all the variables and command window using ‘clear’ and ‘clc’.

Matrices

- **Matrix** – 2D array (table).
  - Elements are ordered in 2 dimensions: rows and columns.
- **M x N matrix** – M rows, N columns.
- Example:
  - myFirstMat = [1 2 3; 4 5 6];

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$



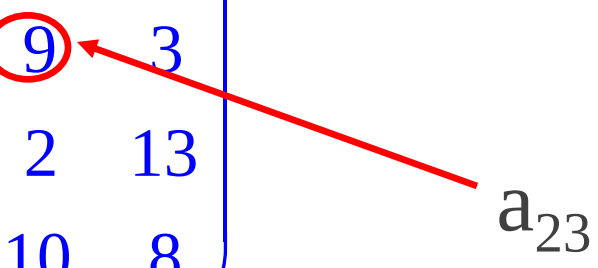
Matrices - indexing

- Indexing: the $\mathbf{a_{ij}}$ element is the element in the **i-th** row and the **j-th** column.

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

- Example:

$$\begin{pmatrix} 5 & 8 & 12 & 4 \\ 7 & 1 & 9 & 3 \\ 11 & 5 & 2 & 13 \\ 3 & 6 & 10 & 8 \end{pmatrix}$$



Arrays

- **Scalar –** 1x1 array.
- **Vector** – **one-dimensional** array.
  - Row: 1 x N array.
  - Column: N x 1 array
- **Matrix** – **two-dimensional** array
  - Table with rows and columns: M x N
- **Three-dimensional** array/matrix – a rectangular cuboid of elements.
  - Dimensions M x N x K.
- **Multi-dimensional** arrays/matrices…

- All these arrays are simply **the same** data-type in Matlab, with just **different dimensions**, or size.

A few more notes about arrays

- Assign values to a variable directly to its place in the array:
  - myVar(2,3) = 5;
  - myVar([1 2],3) = [5 6];
- Assign a value of one variable to another variable:
  - x(2) = y;
- Delete an element from an array:
  - myVec = [1 2 3 4];
  - myVec(2) = [];
  - myVec([2 3]) = [];

Example

- MatlabBasics.m (examples 5-6)

Text variables

- Text variables are comprised of **characters** and marked with **”**.
  - myChar = ‘h’;
  - myChar = ‘5’; (this is not the number 5 but rather the character 5)

- A text variable can contain more than one character → **string** (an array of characters).
  - firstString = ‘hello’;
  - secondString = ‘world’;
  - longerOne = ‘hello world’;
  - longerOne2 = [firstString secondString]; (what’s wrong with that?)

Example

- MatlabBasics.m (example 7)

Practice 2

- In the command window, do the following:
  - Create a text variable that contains one word.
  - Create another text variable that contains one or more words.
  - Concatenate the two strings to create a third variable.
  - Display one of the strings in the command window using ‘disp’ function.

Basic functions

- Matlab has a HUGE number of ready-to-use functions/commands. These are very useful and one of the major advantages of Matlab.
  - Examples: length, size, pwd, clc, clear, disp, sum, mean, std, zeros, rand, randn, save, load, and many more…
- Avoid naming variables/functions with the same name as the basic functions – it temporarily "overrides" them in the current workspace.
  - Tip: Use variable/function names with '_' (vec_size), mix of small and capital letters (vecSize), prefix such as 'my' (myVar), 'this' (thisRun), etc.

Code files

- Matlab code files have a '**.m**' extension.
- They include the lines of code.
- Use the Matlab **editor** to edit and run code files
  - Scripts
  - Functions
  - More on that in the next talk.

Help!

- help *name_of_function*
- lookfor *keyword*
- helpdesk
- Internet

Example

- MatlabBasicsExtra.m (not today…)

Practice 3

- Create a Matlab code file and save it in your current directory. In this file, do the following:
  - Create a 3x4 matrix with values as you like.
  - Change the value of the element in the 2nd row and 3rd column.
  - Change all the values in the 2nd column at once by assigning a new vector.
  - Swap columns 1 and 3.
  - Delete the 4th column.
  - Use ‘size’ function to check for the size of the matrix.