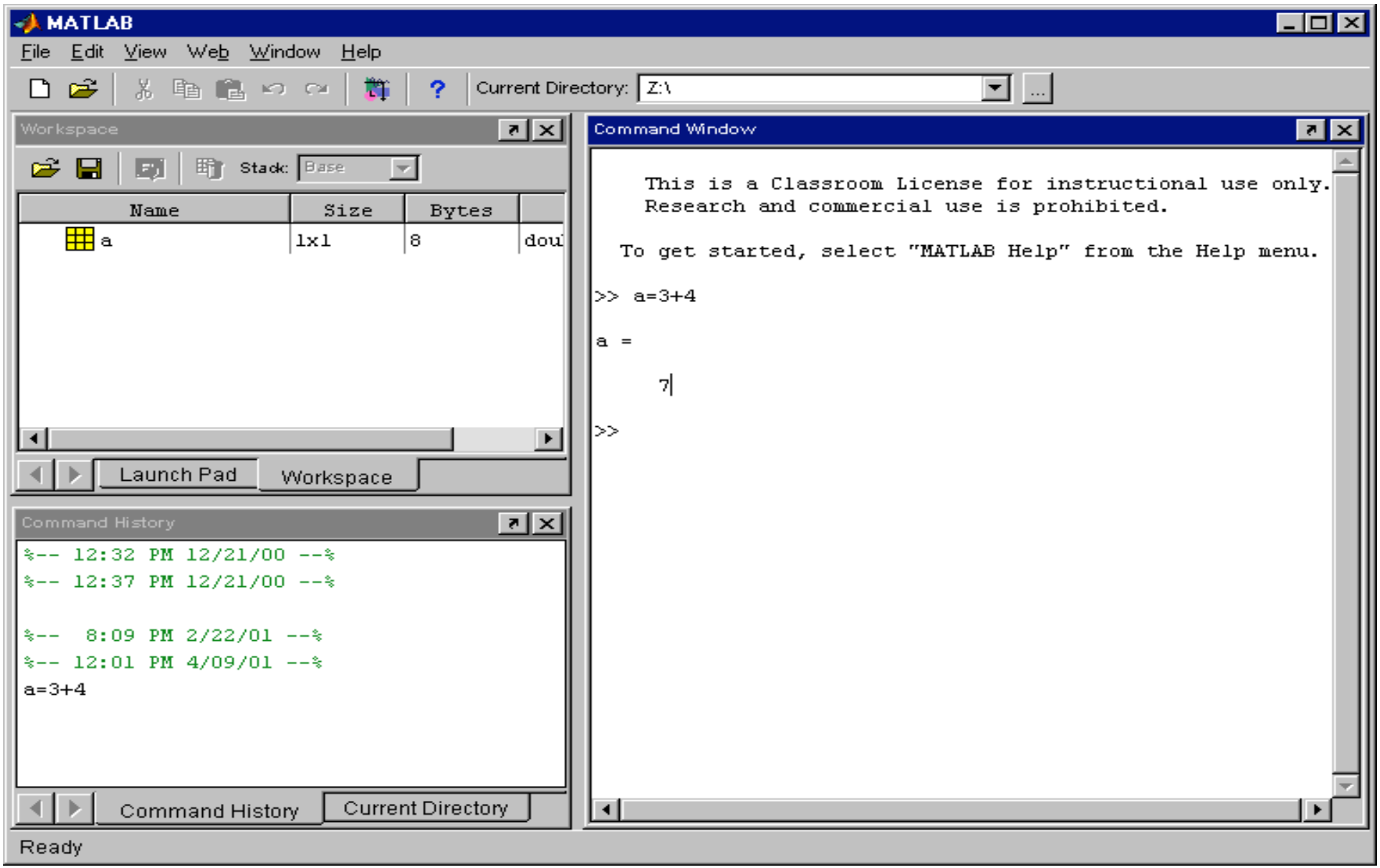
Introduction to Matlab

B.Sc. Sem-4 ELTA Sec-2

What is MATLAB

- MATLAB, short for MATrix LABoratory is a programming package specifically designed for quick and easy scientific calculations and I/O. It has literally hundreds of built-in functions for a wide variety of computations and many toolboxes designed for specific research disciplines, including statistics, optimization, solution of partial differential equations, data analysis.

What You See When you Open MATLAB



- This is the window in which you interact with MATLAB. The main window on the right is called the *Command Window*. You can see the command prompt in this window, which looks like >>. If this prompt is visible MATLAB is ready for you to enter a command. In the figure, you can see that we typed in the command **a=3+4**. In the top left corner you can view the *Launch Pad window* and the *Workspace window*. Swap from one to the other by clicking on the appropriate tag. The *Workspace window* will show you all variables that you are using in your current MATLAB session. In this example, the workspace contains the variable 'a'. When you first start up MATLAB, the workspace is empty. More about workspaces later.
- In the bottom left corner you can see the *Command History window*, which simply gives a chronological list of all MATLAB commands that you used, and the *Current Directory window* which shows you the contents and location of the directory you are currently working in.

- To change the layout of the MATLAB window, select *View*, then *Desktop Layout*. Six different layout styles can be chosen.
- **Note that MATLAB is case-sensitive, so**
- **make sure that the 'Caps Lock' is switched off!**
- During the MATLAB sessions you will create files to store programs or workspaces.
- Create an appropriate folder to store this lab's files.
- Go to the Current Directory window. Find the *Browse for folder* button on the menu (the one with the 3 dots …). Select the folder you just created so that MATLAB will automatically save files in this folder. Note that the current directory (or folder: folder and directory mean the same thing) is also displayed in the top right corner next to the main menu.

Now, let's try a simple command

- Now, let's try a simple command. Next to the prompt in the Command Window type **a=3+4** and then press 'enter' to activate this command. On the screen you should see
- a=
- 7
- Notice how MATLAB carries out this command immediately, and gives you the prompt >> for your next command. Here, a variable called 'a' is created by MATLAB and assigned the value of 7. MATLAB stores the variable a in its *workspace* until you exit MATLAB or tell MATLAB to delete the variable.

- Go to the Workspace window and check that the variable a is in your workspace.
- You will see in this window that a is stored in 8 bytes, that it is a double and that it has size 1x1. This size seems a bit odd and we will explain it later.
- In the Workspace window, double click on the name 'a'. A small window will pop up displaying a’s value. This is a handy feature.
- We can change the value of the variable:

Change the value of the variable:

- We can change the value of the variable:
- Try **a = 9**, followed by 'enter' to change the value of a to 9. Then type **b=sqrt(a)** to find the square root of
  - Don't forget to press 'enter' to enter the command (we will not always remind you to hit 'enter' but it is necessary to tell MATLAB to carry out the command). Now you should see
- b=
- 3
- Go back to the Workspace window and check that b is indeed in your workspace. Also check by double-clicking on a that a's value has changed.

Data representation in MATLAB

- The structure for the storage of all data in MATLAB is a matrix. The MATLAB matrix-variables may have any number of rows and columns. Scalars like the variables a and b that you worked with above are also stored as matrix variables with 1 row and 1 column. This is the reason that the variable a in your workspace is shown as a 1x1 matrix!
- So beware, a matrix-variable can be any variable in MATLAB, that is, it could be a scalar, a vector or a matrix of any size. If we refer to scalars, vectors or matrices specifically we mean just that: scalars, vectors or matrices.

Entering variables

An mxn ('*m* by *n*') MATLAB matrix-variable (or simply variable) has *m* rows and *n* columns. For example, the

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

is a 2x3 *matrix*. The numbers 1 through 6 are called the *elements* of the matrix. The element on row 1 and column 2 has the value 2. The element on row 2 and column 3 has the value 6.

Entering variables

- Create the matrix A by typing A = [1 2 3;4 5 6] followed by the enter. Make sure that you separate the elements 1,2 and 3, and 4, 5 and 6 with a space. If you don't MATLAB will think you created the vector A with first element 123 (one hundred and twenty-three) and second element 456. Also, use *square* brackets instead of parentheses.
- Instead of a space, you can also use a comma to separate the elements. The semi-colon (;) in this context is used to separate the elements of one row from another, but you can also use a line break to separate rows as shown below.
-

  Remove the matrix A by typing clear A. Then recreate A using
- >> A = [1,2,3 *(now hit enter for the line break)*
- 4,5,6]

Error messages

- If your command is invalid MATLAB gives you explanatory error messages. Read them carefully. Normally MATLAB points to the exact position of where things went wrong.

  Try to change A to $\begin{bmatrix} 11 & 22 \\ 33 & 44 \end{bmatrix}$

  using the incorrect command **A = [11 22 33;44]**

  The command has failed. MATLAB tells you that you do not have the same number of columns in each row, i.e. the same number of elements in each row. You put the semi-colon in the wrong place.

Row and column vectors

- A row vector has one row and any number of columns. Similarly a column vector has one column and any number of rows.
- Type **v=[1, 2, 3]** . Now use the **size** function to check that v has 1 row and 3 columns by typing **n= size(v)**
- **size** is an example of a MATLAB function. The round brackets enclose the variable to be operated on. The first number returned by the **size** function gives the number of rows in the variable and the second number gives the number of columns.

Row and column vectors

Enter the column vector $\mathbf{c} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

using **c = [1; 2; 3]** and check that is has 3 rows and 1 column (note the use of semicolons to separate the rows).

Changing matrices

Just as with scalars, you can change a matrix simply by defining it to be something else. We already unsuccessfully tried this when we tried to change A. Lets get it right this time.

Type **A=[11 22;33 44]** to make A become the 2x2 matrix given earlier.

The 2x3 matrix that A used to be has now been lost. The variable A still exists of course in MATLABs workspace but now has a *different size and different values for its elements*.

Scrolling

- Suppose you want to repeat or edit an earlier command. If you dislike typing it all back in, then there is good news for you: MATLAB lets you search through your previous commands using the up-arrow and down-arrow keys. This is a very convenient facility, which can save a considerable amount of time.
- Suppose you regret the last command, and that you want A still to be the 2x3 matrix you originally defined. Press the up-arrow key until the earlier command
- **A=[1 2 3;4 5 6]** appears in the Command Window (the down-arrow key can be used if you go back too far). Now press enter. The matrix A is back to what it was originally.

Scrolling

- You can speed up the scroll if you remember the first letters of the command you are interested in. For example, if you quickly want to jump back to the command **v = [1, 2, 3]** just type **v =** and then press the up-arrow-key.
- MATLAB will display the most recent command starting with 'v ='. Alternatively you can use 'Copy' and 'Paste' under 'Edit'.
-
- In MATLAB 6 you can also use the Command History window to jump to a previous command:
-
- Go to the Command History window. Jump back to the command **v=[1,2,3]** by double-clicking on this line in the Command History window. The command will be carried out immediately.