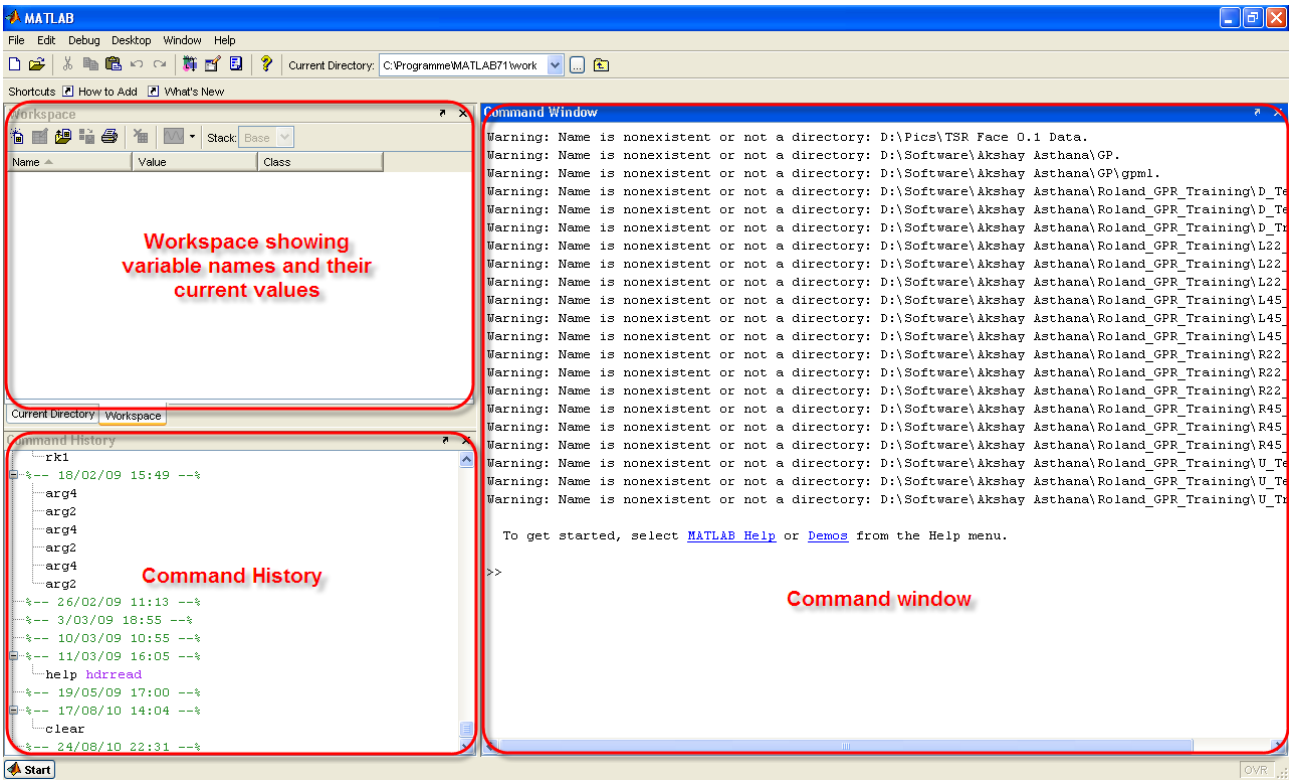
Tutorials and Laboratories

Week 1

The purpose of this week's tutorial and lab exercises is to introduce you to the Matlab programming and experimental environment, which we will use extensively throughout the unit. Matlab is an extremely powerful toolbox. Please note that this week's class on Matlab can really only be a first introduction to Matlab. We will introduce and demonstrate new commands throughout the unit. Today, we simply want to cover some of the basics. Further tutorials on Matlab are available at http://www.mathworks.com/academia/student_center/tutorials/launchpad.html (see below).

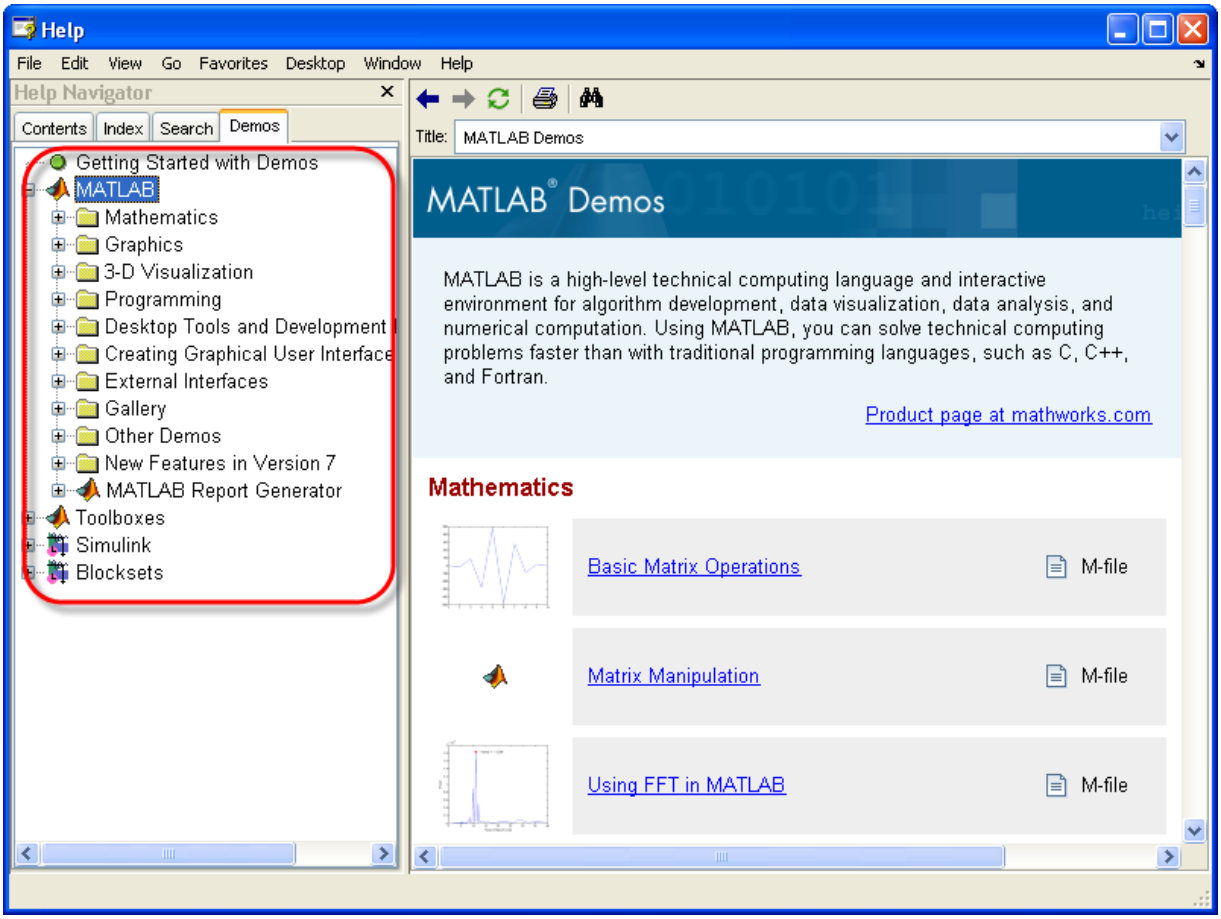
Matlab is proprietary software but we have a campus wide licence that allows students to install it on their own computers as well (see this unit's UC Learn site for further information). Alternatively, the open-source community has written a Matlab 'clone' called Octave, which comes under the GNU General Public Licence and can therefore be downloaded and installed for free. 99.9% of Matlab code runs on Octave and vice versa.

1. See if you can find Matlab under Start → All Programs. If you have trouble locating it, try typing in "Matlab" in the search textbox or ask your tutor for help.
2. You will see a window looking something like this (it might look slightly different, depending on the version of Matlab):



The so-called *Workspace* in the top-left shows variable names and their current values. Note that such variables can be vectors or matrices.
The *Command History* window shows the most recent commands.
The *Command Window* is where you type in command-line style commands.

3. Matlab comes with its own built-in demo system. In the Command Window, type in `demo`, then hit enter. You will see a new window popping up looking something like this



You can select a demo on the right, or choose one from the MATLAB folder on the left.

View the "Getting Started with MATLAB" video first.

Then, view the "Working in the Development Environment" video.

4. To get help on a topic, you can start the built-in helpdesk system by simply typing in 'helpdesk'. A new window will pop up where you can search for the right answer on the topic.

   Alternatively, simple type in `help` to give you a list of available help topics to choose from, or, if you know the topic, you can simply type `help topic`.

5. Type in `help ops` to view a list of operators. Each operator can be shown in more detail by typing in `help operatorName`.

6. The **Matlab Help** (http://www.mathworks.com/help/techdoc/index.html) is a great resource! Explore the Getting Started section. It includes some short videos that you should view.

7. In its simplest form, Matlab works like a calculator. Try typing in

   ```
   2 + 3
   ```

   and hit the enter key. What happens?

8. Unlike most other programming languages, Matlab lets you get away with not having to declare a variable's type before the first use. Matlab figures it out from the first time you use that variable. Hence, we can simply write

```
x = 2 + 3
```

and after hitting the enter key, we see that the command has been executed and that the value of x is now 5. You should also be able to see x being listed in the Workspace subwindow.

9. If you want to assign a new value to x, simply overwrite it. That is, if we want to set it to, say, 10, we simply write

```
x = 10
```

10. If we want to remove a particular variable from the *Workspace* subwindow, we simply write

```
clear variableName
```

where the latter represents the name of the variable. Note how it is removed from the Workspace window when you hit the enter key. If you want to remove all variables from the Workspace, type in `clear all`. It is often a good idea to start a Matlab program with a `clear all` command to clear the Workspace from any remnants of previous variables.

11. The letter 'i' is used to describe the imaginary part of a complex number (what that is will be explained in one of the next lectures). Try the following commands in the Command window

```
>> complex_num1 = 1 + 2*i
```

will give you

```
complex_num1 =
   1.0000 + 2.0000i
```

One of the funny things about complex numbers is that this 'i' is defined as the square root of -1

```
>> complex_num2 = sqrt(-1)
```

will give you

```
complex_num2 =
        0 + 1.0000i
```

Multiplying these two numbers results in

```
>> complex_num1 * complex_num2
ans =
  -2.0000 + 1.0000i
```

12. Matlab works a lot with functions, either built-in ones or the ones you defined yourself. For example, the function `abs()` gives you the absolute value of the input

```
>> abs(-2)
ans =
     2
```

13. Comments in Matlab use the % sign.

14. Ending a command with a semi-colon means that the output is not shown on the screen.

15. Download the example image from the Moodle site.
16. See if you can work out how to load it in Matlab. Load the image into a variable called `myImg`.

    Hint: Explore the `imread` command. Remember Matlab's online documentation (http://www.mathworks.com/help/techdoc/index.html). Specific information for loading images can be found at http://www.mathworks.com/help/techdoc/ref/f16-5702.html#f16-7535.

    How can you display the image? Hint: Explore the `figure` and `imshow` commands.

    Note: The online documentation is a great resource. Scroll down to the end of the description for a particular command and you will always see some examples being given. This is a great way of learning how to work with Matlab. Get used to working with the online documentation, as it will help you tremendously with the practical work in this unit.
17. Explore the use of the `imresize` command. Again, look it up in the online Matlab help or type in the command window `help imresize`.

    How can you resize an image, for example your `myImg`, to half the size?

18. In Matlab, an image is simply stored as a 2-dimensional **matrix**. To find the size of a matrix (or an image), use the `size` command.

    Type in `size(myImg)` and hit enter. Observe what happens. What does the result mean?

19. A specific form of a matrix is a 1-dimensional matrix, more commonly known as a **vector**. Matlab knows about row vectors and column vectors.

    Type in `x = [1 23 -456 66]` (note the spaces between the numbers). This creates a **row vector**. Use the size command to view the size of this vector.

20. Now, type in `y = [1; 23; -456; 66]` . This creates a **column vector**. Again, use the size command to view the size of this vector.

21. To change a row vector into a column vector and vice versa, mathematics knows the concept of a **transpose**. In Matlab, this is done by appending the apostrophe ' to a vector.

    Type in `x'` and observe what happens. What happens if you type in `size(x')` ?

22. So far, we have worked directly on the Matlab command line. If we want to run the same code again, we can use the up and down arrow keys to get to the same command again, but this is not very useful when we want to write a bigger, more complex program. Luckily, Matlab knows script files, so called .m files, that allows to put commands together and then to run this script.

    Watch the "Writing a MATLAB program" video from the Matlab Demos window.
23. In the Matlab window, click on File -> New -> Script. A new window will pop up. Copy the commands you have written so far into the window. Use % to enter comments in between command lines to separate them and to remind you what the commands were used for.

24. Save the Matlab script file as `myFirstMatlabProgram.m` on your USB drive. Go back to the command line environment and type in `myFirstMatlabProgram`. What happens?

25. Write a new Matlab program. Call it `imageColourConversion.m`. In this program, load the image (the same as before or any other one you might have handy). Display the image in figure 1.

    Resize the image to a quarter of the original size. Store this quarter size image in a new variable. Display it in figure 2.

    Separate the three colour components Red, Green and Blue and display them in separates figures (3, 4 and 5, respectively).

    Now convert the original colour image to grayscale by using the `rgb2gray` command. Display this in figure 6. How does this one differ from the red, green and blue images?

Additional Video Resources

1. Up and running with Matlab [LINK]
2. Up and running with Octave [LINK]