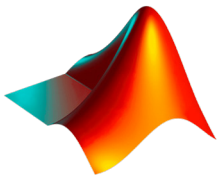
A BEGINNER'S GUIDE

to



MATLAB®

Jack Lebedev
Department of Computer Science
Darwin University

Table of Contents

1. INTRODUCTION

1.1 MATLAB at Darwin University 3
1.2 How to read this tutorial 3

2. MATLAB BASICS

2.1 The basic features 4
2.2 Vectors and matrices 5
2.3 Built-in functions 12
2.4 Plotting 21

3. PROGRAMMING IN MATLAB

3.1 M-files: Scripts and functions 27
3.2 Loops 29
3.3 If statement 32

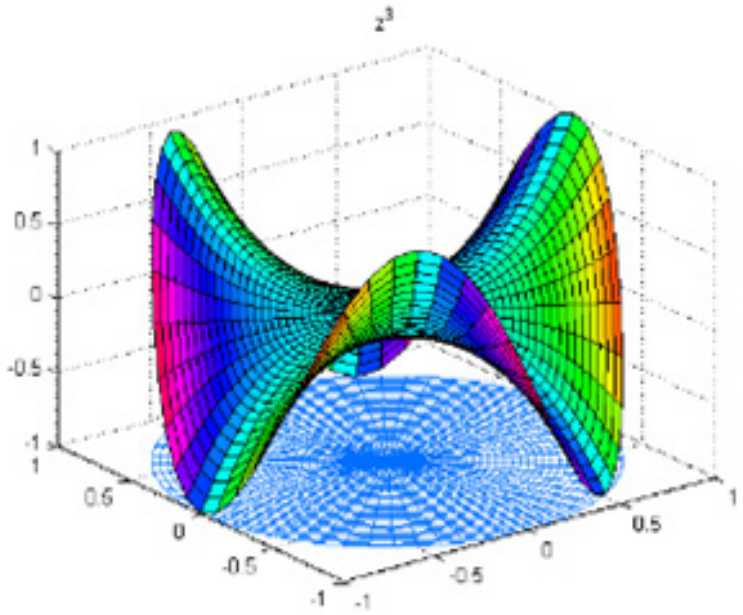
4. ADDITIONAL TOPICS

4.1 Polynomials in MATLAB 35
4.2 Numerical Methods 37

5. CLOSING REMARKS AND REFERENCES 41

1. INTRODUCTION

Short for MATrix LABoratory, MATLAB is a revolutionary software for mathematical modelling, widely used both in academic as well as workplace settings. This interactive self-contained program is used to create numerical computation and data visualizations, while programming features used in conjunction produce an incredible tool for countless engineering and scientific tasks. A notable difference between other mathematic software systems (ex. Wolfram's Mathematica or Maplesoft) & MATLAB, is that the latter requires Toolboxes to perform symbolic computations (commonly seen in discrete mathematics for example). It still however boasts an extensive array of capabilities in numeric computations. One of MATLAB's most popular capabilities are its matrice-generating functions. A 1-by-1 matrice would be a scalar, and a vector with a row containing 5 values would be classed as a 1-by-5 matrix. This category along with numerous other features of the software will be expanded on in the rest of this guide to MATLAB. One of the key advantages present in MATLAB, making the program easier to use than its competitor applications, is its extensive use of natural numbers in notation (think along the lines of what we encounter in linear algebra). This lends smoothness to the user experience, as well as a shorter learning curve, making MATLAB a superior option for computations with natural numerics. After all, the main purpose of our guide is to help beginners with a smooth transition to MATLAB, through exploration of its most popular basic commands and features. It does not cover the entirety of the software's features, and students are encouraged to explore further options in MATLAB by looking into other suggested sources of continued learning, present at the end of the guide.



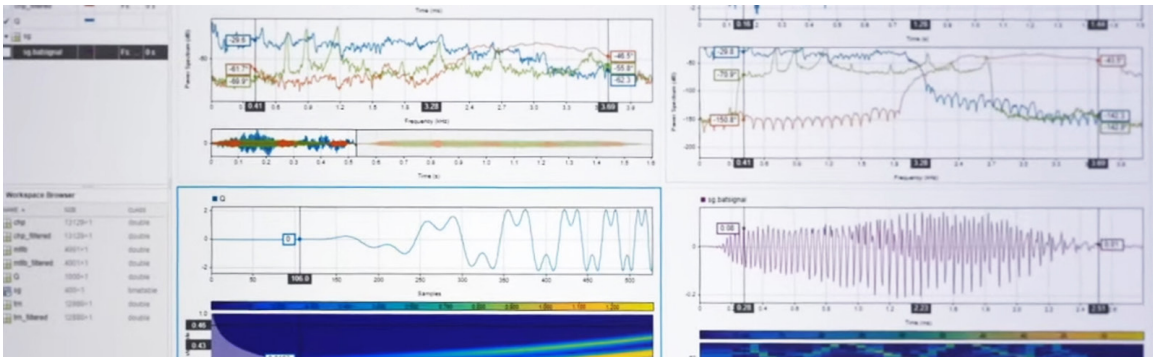
1.1 MATLAB At Darwin University

To begin with, let us give the good news that MATLAB can be run on literally any networked PC, and can be accessed on the MetaFrame Presentation Server (**http://www.darwin.edu/moresoftware/matlab**). To gain access you must log in with your student username and password. Once logged in, a folder with applications marked “MATLAB” should be double-clicked and the program will launch.

Note: For first-time users, some client software may have to be installed on your computer.

Next, simply follow the prompts on the webpage post-login, and the program will launch in a separate window. Follow the “begin” prompt and arrow. By default, the working sub-directory is **D:\Applications\matlabR20**. Do not save any work in this default directory. Save it instead to the **G:** drive containing your account, with command >> cd **G:** (from inside MATLAB). Refer to your instructor for further guidance on how or where to save any work.

When done using MATLAB, it is important to log out of the Meta Frame Presentation Server.



MATLAB’s vast library of toolboxes

1.2 How to read this guide

When referencing the sections to follow, remember that a MATLAB prompt **(»)** will indicate a command has been entered. What follows after this point indicates user input (known as a “command”), followed by the carriage return (i.e., the “Enter” key). Such inputs are generally followed by outputs, and unless stated otherwise, a line or lines following a command denotes “output” (i.e., the MATLAB-generated result based on what a user typed in). Remember that this software is case-sensitive, meaning for example **‘a + B’** is not identical to **‘a + b’**. Varying fonts will also be used to simulate an interactive session. See the example below:

Since MATLAB works as a calculator, when we add 2 numbers we can expect the following result:

```
» 3 + 4
ans = 7
```

MATLAB is however, far more useful than serving as a simple “calculator”. To take full advantage of this guide, we strongly advise the reader to self practice all the commands taught in each section, then work on all the provided exercises. It is easier to go through this guide once for a general overview, then read it again (possibly even again and again) whilst in front of your computer.

2. MATLAB BASICS

2.1 Basic features

To start, let's take a look at a row vector containing values 1 to 5, assigning it the variable name "x":

```
» x = [1 2 3 4 5]

x = 1 2 3 4 5
```

Pay attention to how the equal sign is used to assign the name "x" to the vector, entries are ordered inside brackets, and each value is separated by a space (very similar to a linear algebra notation). You might've seen commas **(,)** used to separate each value in other programs, instead of spaces, or some combination of both. Using either spaces or commas in MATLAB provides a huge difference in your command results! For generating a column vector (there's a difference between row and column vectors, naturally), use either semicolons **(;)** to separate each entry or establish a row vector, then take the "transpose" to generate a column vector. In the next example, we'll achieve this by creating the column vector "y" which will contain values from 6 to 10, displaying both techniques:

```
» y = [6;7;8;9;10]

y =
6
7
8
9
10
```

```
» y = [6,7,8,9,10]

y = 6 7 8 9 10
» y'
ans =
6
7
8
9
10
```

A few observations from above: first, see how the transpose of a vector (or matrix) uses a single quote **(')**. After processing this, MATLAB will repeat what was typed. If you do not want to display the output of a command, use a semicolon **(;)** at the end of the command line. Now, keep in mind that the variable name "ans" will be auto assigned by MATLAB if you did not manually assign a name. Referencing the above example, which would result in a new variable getting generated with its value being the column vector entries. Therefore "ans" becomes re-assigned any time you type a command without appointing a variable(s).

It's traditionally done as due diligence to track all the defined variables – but since this gets troublesome, MATLAB does it for us. There's a command "whos" which reports which variables are active:

```
» whos
  Name        Size            Elements    Bytes    Density   Complex

  ans         5 by 1                 5       40       Full        No
  x           1 by 5                 5       40       Full        No
  y           1 by 5                 5       40       Full        No

Grand total is 15 elements using 120 bytes
```

Another somewhat similar command, "who", will tell you the names of all active variables.