A MATLAB Tutorial

Ed Overman
Department of Mathematics
The Ohio State University

(August 12, 2016 2:07 p.m.)

Introduction **3**
1 Scalar Calculations **6**
1.1 Simple Arithmetical Operations 6
1.2 Variables 7
1.3 Round-off Errors 9
1.4 Formatting Printing 10
1.5 Common Mathematical Functions 11
1.6 Complex Numbers 13
1.7 Script M-files 14
1.8 Help! 15
1.9 Be Able To Do 17
2 Arrays: Vector and Matrix Calculations **17**
2.1 Generating Matrices 18
2.2 The Colon Operator 22
2.3 Manipulating Vectors and Matrices 23
2.4 Simple Arithmetical Operations 28
2.5 Operator Precedence 31
2.6 Be Careful! 32
2.7 Common Mathematical Functions 34
2.8 Data Manipulation Functions 34
2.9 Advanced Topic: Multidimensional Arrays 37
2.10 Be Able To Do 38
3 Anonymous Functions, Strings, and Other Data Types **39**
3.1 Anonymous Functions 40
3.2 Passing Functions as Arguments 41
3.3 Strings 42
3.4 Cell Arrays and Structures 43
3.5 Advanced Topic: Data Types and Classes 46
3.6 Be Able To Do 48
4 Graphics **49**
4.1 Two-Dimensional Graphics 49
4.2 Three-Dimensional Graphics 59
4.3 Advanced Topic: Functions 61
4.4 Advanced Topic: Handles and Properties 66
4.5 Advanced Topic: GUIs (Graphical User Interfaces) 68
4.6 Advanced Topic: Making Movies 74
4.7 Be Able To Do 77
5 Solving Linear Systems of Equations **77**
5.1 Square Linear Systems 77
5.2 Catastrophic Round-Off Errors 80
5.3 Overdetermined and Underdetermined Linear Systems 81
6 File Input-Output **83**
7 Some Useful Linear Algebra Functions **84**

8 Programming in MATLAB . . . **91**
8.1 Control Flow and Logical Variables . . . 91
8.2 Matrix Relational Operators and Logical Operators . . . 95
8.3 Function M-files . . . 99
8.4 Odds and Ends . . . 111
8.5 Advanced Topic: Vectorizing Code . . . 113
9 Sparse Matrices . . . **116**
10 Initial-Value Ordinary Differential Equations . . . **119**
10.1 Basic Functions . . . 119
10.2 Advanced Functions . . . 124
11 Boundary-Value Ordinary Differential Equations . . . **130**
12 Polynomials and Polynomial Functions . . . **133**
13 Numerical Operations on Functions . . . **136**
14 Discrete Fourier Transform . . . **139**
15 Mathematical Functions Applied to Matrices . . . **146**
Appendix: Reference Tables . . . **149**
Arithmetical Operators . . . 149
Special Characters . . . 149
Getting Help . . . 150
Predefined Variables . . . 150
Format Options . . . 150
Some Common Mathematical Functions . . . 151
Input-Output Functions . . . 152
Arithmetical Matrix Operations . . . 152
Elementary Matrices . . . 153
Specialized Matrices . . . 153
Elementary Matrix Operations . . . 153
Manipulating Matrices . . . 154
Odds and Ends . . . 154
Two-Dimensional Graphics . . . 155
Three-Dimensional Graphics . . . 155
Advanced Graphics Features . . . 156
String Functions, Cell Arrays, Structures, and Classes . . . 157
Data Manipulation Functions . . . 157
Some Useful Functions in Linear Algebra . . . 158
Logical and Relational Operators . . . 158
Control Flow . . . 158
Logical Functions . . . 159
Programming Language Functions . . . 160
Debugging Commands . . . 160
Discrete Fourier Transform . . . 160
Sparse Matrix Functions . . . 161
Time Evolution ODE Solvers . . . 161
Boundary-Value Solver . . . 161
Numerical Operations on Functions . . . 161
Numerical Operations on Polynomials . . . 162
Matrix Functions . . . 162
Solutions To Exercises . . . **163**
ASCII Table . . . **167**
Index . . . **169**

Introduction

MATLAB is an interactive software package which was developed to perform numerical calculations on vectors and matrices. Initially, it was simply a MATrix LABoratory. However, today it is much more powerful:

- It can do quite sophisticated graphics in two and three dimensions.
- It contains a high-level programming language (a "baby C") which makes it quite easy to code complicated algorithms involving vectors and matrices.
- It can numerically solve nonlinear initial-value ordinary differential equations.
- It can numerically solve nonlinear boundary-value ordinary differential equations.
- It contains a wide variety of toolboxes which allow it to perform a wide range of applications from science and engineering. Since users can write their own toolboxes, the breadth of applications is quite amazing.

Mathematics is the basic building block of science and engineering, and MATLAB makes it easy to handle many of the computations involved. You should not think of MATLAB as another complication programming language, but as a powerful calculator that gives you fingertip access to exploring interesting problems in science, engineering, and mathematics. And this access is available by using only a small number of commands and functions[†] because MATLAB's basic data element is a matrix (or an array).

This is a crucial feature of MATLAB — it was designed to group large amounts of data in arrays and to perform mathematical operations on this data as individual arrays rather than as groups of data. This makes it very easy to apply complicated operations to the data, and it make it very difficult to do it wrong. In high-level computer languages you would usually have to work on each piece of data separately and use loops to cycle over all the pieces. In MATLAB this can frequently do complicated "things" in one, or a few, statements (and no loops). In addition, in a high-level language many mathematical operations require the use of sophisticated software packages, which you have to find and, much worse, to *understand* since the interfaces to these packages are frequently quite complicated and the documentation must be read and mastered. In MATLAB, on the other hand, these operations have simple and consistent interfaces which are quite easy to master. For an overview of the capabilities of MATLAB, type

```
>>  demo
```

in the `Help Navigator` and click on `MATLAB`.

This tutorial is designed to be a concise introduction to many of the capabilities of MATLAB. It makes no attempt to cover either the range of topics or the depth of detail that you can find in a reference manual, such as *Mastering MATLAB 7* by Duane Hanselman and Bruce Littlefield (which is over 850 pages long) or *MATLAB Guide, 2nd edition* by Desmond and Nicholas Higham (which is almost 400 pages long). This tutorial was initially written to provide students with a *free* "basic" overview of functions which are useful in an undergraduate course on linear algebra. Over the years it has grown to include courses in ordinary differential equations, mathematical modelling, and numerical analysis. It also includes an introduction to two- and three-dimensional graphics because graphics is often the preferred way to present the results of calculations.

In this tutorial MATLAB is first introduced as a calculator and then as a plotting package. Only afterwards are more technical topics discussed. We take this approach because most people are quite familiar with calculators, and it is only a small step to understand how to apply these same techniques to matrices rather than individual numbers or varibles. In addition, by viewing MATLAB as a simple but powerful calculater, rather than as a complicated software package or computer language, you will be in the correct frame of mind to use MATLAB.

You should view MATLAB as a tool that you are "playing with" — trying ideas out and seeing how they work. If an idea works, fine; if it doesn't, investigate further and figure out why. Maybe you misunderstood some MATLAB command/function, or maybe your idea needs some refinement. "Play around"

[†]There is a technical distinction between a *command* and a *function* in MATLAB: input arguments to commands are not enclosed in parentheses (they are separated by spaces) and there are no output arguments (i.e., a command cannot be on the right-hand side of an equal sign). In reality, this is a very fine distinction since many commands can be written as functions by putting the arguments between parentheses and separating them with commas. We will generally use the terms interchangably.

interactively and figure it out. There are no hard and fast rules for figuring it out — try things and see what happens. Don't be afraid to make mistakes; MATLAB won't call you an idiot for making a mistake. When you first learned to ride a bicycle, you fell down a lot — and you looked pretty silly. But you kept at it until you didn't fall down. You didn't study Newton's laws of motion and try to analyze the motion of a bicycle; you didn't take classes in how to ride a bicycle; you didn't get videos from the library on how to ride a bicycle. You just kept at it, possibly with the assistance of someone who steadied the bicycle and gave you a little push to get you started. This is how you should learn MATLAB.

However, this tutorial is not designed for "playing around". It is very ordered, because it has been designed as a brief introduction to all the basic topics that I consider important and then as a reference manual. It would be very useful for students to have a document which uses this "play around" approach so you would learn topics by using them in exploring some exercise. This is how workbooks should be written: present some exercise for students to investigate, and let them investigate it themselves. And these exercises should be interesting, having some connection to physical or mathematical models that the students — or at least a reasonable fraction thereof — have some knowledge of and some interest in. This tutorial is designed to be a reference manual that could be used alongside such a workbook — if only someone would write it.

Summary of Contents

We have tried to make this tutorial as linear as possible so that the building blocks necessary for a section are contained in preceding sections. This is not the best way to learn MATLAB, but it is a good way to document it. In addition, we try to separate these building blocks and put them in short subsections so that they are are easy to find and to understand. Next, we collect all the commands/functions discussed in a subsection and put them in a box at the end along with a very brief discussion to make it easy to remember these commands. Finally, we collect them all and put them in the appendix, again boxed up by topic. MATLAB has a **HUGE** number of commands/functions and this is one way to collect them for easy reference.

Warning: Usually we do not discuss the complete behavior of these commands/functions, but only their most "useful" behavior. Typing

```
>>  help <command/function>
```

or

```
>>  doc <command/function>
```

gives you complete information about the command/function.

Notation: `help <command/function>` means to enter whatever command/function you desire (without the braces).

`help command/function` means to type these two words as written.

Section 1 of this tutorial discusses how to use MATLAB as a "scalar" calculator, and Section 2 how to use it as a "matrix" calculator. Following this, you will be able to set up and solve the matrix equation $\mathtt{Ax} = \mathtt{b}$ where $\mathtt{A}$ is a square nonsingular matrix.

Section 4 discusses how to plot curves in two and three dimensions and how to plot surfaces in three dimensions. These three sections provide a "basic" introduction to MATLAB. At the end of each of these three sections there is a subsection entitled "Be Able To Do" which contains sample exercises to make sure you understand the basic commands/functions discussed. (Solutions are included.)

You have hopefully noticed that we skipped section 3. It discusses a number of minor topics. Since they are useful in generating two- and three-dimensional plots, we have included it here.

The following sections delve more deeply into particular topics. Section 5 discusses how to find any and all solutions of $\mathtt{Ax} = \mathtt{b}$ where $\mathtt{A} \in \mathbb{C}^{m\times n}$ need not be a square matrix; there might be no solutions, one solution, or an infinite number to this linear system. When no solution exists, it discusses how to calculate a least-squares solution (i.e., the "best" approximation to a solution). In addition, it discusses how round-off errors can corrupt the solution, and how to determine if this is likely to occur.

Section 6 is quite brief and discusses advanced functions to input data into MATLAB and output it to a file. (The basic functions are discussed in Section 4.1.) This is useful if the data is being shared between various computer programs and/or software packages.

Section 7 discusses a number of functions which are useful in linear algebra and numerical linear algebra. Probably the most useful of these is calculating some or all of the eigenvalues of a square matrix.

Section 8 discusses MATLAB as a programming language — a very “baby C”. Since the basic data element of MATLAB is a matrix, this programming language is *very* simple to learn and to use. Most of this discussion focuses on writing your own MATLAB functions, called function m-files (which are similar to functions in C and to functions, more generally subprograms, in Fortran). Using these functions, you can code a complicated sequence of statements such that all these statements as well as all the the variables used by these functions are hidden and will not affect the remainder of your MATLAB session. The only way to pass data into and out of these functions is through the argument list.

Section 9 discusses how to generate sparse matrices (i.e., matrices where most of the elements are zero). These matrices could have been discussed in Section 2, but we felt that it added too much complexity at too early a point in this tutorial. Unless the matrix is **very large** it is usually not worthwhile to generate sparse matrices — however, when it is worthwhile the time and storage saved can be boundless.

Section 10 discusses how to use MATLAB to numerically solve initial-value ordinary differential equations. This section is divided up into a “basic” part and an “advanced” part. It often requires very little effort to solve even complicated odes; when it does we discuss in detail what to do and provide a number of examples. Section 11 discusses how to use MATLAB to numerically solve boundary-value ordinary differential equations.

Section 12 discusses how to numerically handle standard polynomial calculations such as evaluating polynomials, differentiating polynomials, and finding their zeroes. Polynomials and piecewise polynomials can also be used to interpolate data.

Section 13 discusses how to numerically calculate zeroes, extrema, and integrals of functions.

Section 14 discusses the discrete Fourier transform and shows how it arises from the continuous Fourier transform. We also provide an example which shows how to recover a simple signal which has been severely corrupted by noise.

Finally, Section 15 discusses how to apply mathematical functions to matrices.

There is one appendix which collects all the commands/functions discussed in this tutorial and boxes them up by topic. If a command/function has more than one use, it might appear in two or more boxes.

This tutorial closes with an index. It is designed to help in finding things that are “just on the tip of your tongue”. All the MATLAB commands/functions discussed here are listed at the beginning of the index, followed by all the symbols, then followed by a list of all the script and function m-files which are in the companion zip file. Only then does the alphabetical index begin (which contains all of them again).

Notation: A variable, such as `x`, can represent any number of types of data, but usually it represents a scalar, a vector, or a matrix. We distinguish them by using the lowercase `x` when it is a scalar or a vector, and the uppercase `X` when it is a matrix. Also, in MATLAB vectors can be either row vectors, e.g., $(1,2,3)$ or column vectors $(1,2,3)^{\mathrm{T}}$ (where “$^{\mathrm{T}}$” is the transpose of a vector or matrix). In a linear algebra setting we always define `x` to be a column vector. Thus, for example, matrix-vector multiplication is always written as `A*x` and the inner product of the two vectors `x` and `y` is `x'*y`, i.e., $x_1y_1 + x_2y_2 + \cdots + x_ny_n$ (where “`'`” is the MATLAB symbol to take the transpose of a real vector or matrix).