Chapter 1

Introduction

MATLAB (Matrix laboratory) is an interactive software system for numerical computations and graphics. As the name suggests, MATLAB is especially designed for matrix computations: manipulating arrays, solving systems of linear equations, computing eigenvalues and eigenvectors, factoring matrices, and so forth. In addition, it has a variety of graphical capabilities: plotting data, creating animations, creating user interfaces, and can be extended through programs written in its own programming language.

MATLAB is designed to solve problems numerically, that is, in finite-precision arithmetic. Therefore it produces approximate rather than exact solutions, and should not be confused with a symbolic computation system (SCS) such as Mathematica or Maple. It should be understood that this does not make MATLAB better or worse than an SCS; it is a tool designed for different tasks and is therefore not directly comparable.

In the following sections, we give an introduction to some of the most useful features of MATLAB. We include plenty of examples; the best way to learn to use MATLAB is to read this while running MATLAB, trying the examples and experimenting.

1.1 Starting MATLAB
Microsoft Windows: Double-click on the MATLAB shortcut icon on the Windows desktop, or find it under Start - Programs - Math & Numerical Methods - Matlab 6.5 - Matlab 6.5
Linux: Type MATLAB at the operating system prompt.

1.2 Quitting MATLAB
To end your MATLAB session, select Exit MATLAB from the File menu in the desktop, or type quit in the Command Window. To execute specified functions each time MATLAB quits, such as saving the workspace, you can create and run a finish.m script.

1.3 MATLAB Environment
When you start MATLAB, a special window called the MATLAB desktop appears, Figure 1.1. It integrates many tools for managing files, variables, applications, and debugging within the MATLAB environment. This is what is known as an integrated development environment (IDE).
1.4 The Desktop
The major tools within or accessible from the desktop are:

<table>
<thead>
<tr>
<th>Window</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Window</td>
<td>Main window: for running functions and entering variables</td>
</tr>
<tr>
<td>Command History</td>
<td>Logs previously run commands in the Command Window</td>
</tr>
<tr>
<td>Launch Pad</td>
<td>Provides access to tools, demos, and documentation</td>
</tr>
<tr>
<td>Help Browser</td>
<td>Provides help for accessing documentation</td>
</tr>
<tr>
<td>Current Directory</td>
<td>Browser for accessing files</td>
</tr>
<tr>
<td>Workspace</td>
<td>Provides information about the variables that are used</td>
</tr>
<tr>
<td>Editor/Debugger</td>
<td>Helpful in creating, modifying, and debugging MATLAB script and function files (M-files)</td>
</tr>
<tr>
<td>Figure Window</td>
<td>Contains output from graphic commands</td>
</tr>
<tr>
<td>Array Editor</td>
<td>Browser for editing arrays</td>
</tr>
</tbody>
</table>

1.4.1 Command Window
Use the command window to enter variables and run functions and M-files. Type the functions and variables at the MATLAB prompt (>>) and they will be executed on the spot
1.4.2 Notes on working with the Command Window

- To type a command, the cursor MUST be placed next to the command prompt “>>”
- The end of a command is when the Enter key is pressed. Once a command is entered, it is executed. However, only the last command is executed. Previous commands are unaltered and unexecuted.
- Several command may be typed on the same line by typing a comma “,” between the commands. Again, when the Enter key is pressed, the commands are executed in order from left to right.
- A previous command may be “recalled” to be the current command by using the up arrow. When the command is displayed at the command prompt, it can be modified is needed and then executed. The down-arrow may be used to move down the previously typed commands.
• If a command is too long to fit on one line, it can be split between lines by typing three periods “...”, pressing the Enter key, and continuing the command on the next line. A command may extended over several lines up to a total of 4096 characters.

1.4.3 Continuation and Suppress Echo:

<table>
<thead>
<tr>
<th>MATLAB</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statement: Terminated by an end-of-line &lt;carriage return&gt; unless you put a ... at the end of the line to continue. Can also terminate with a semicolon ; which suppresses the echo of the statement.</td>
<td>Statement: Terminated with a semicolon</td>
</tr>
<tr>
<td>x1 = 1 + 1/2 + 1/3 + 1/4 ... + 1/5 + 1/6;</td>
<td>x1 = 1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6;</td>
</tr>
<tr>
<td>Continuation: ...</td>
<td>NA</td>
</tr>
<tr>
<td>Suppress echo: ;</td>
<td>NA</td>
</tr>
<tr>
<td>&gt;&gt; x = 3 %evaluation printed</td>
<td></td>
</tr>
<tr>
<td>x = 3</td>
<td>x = 3;</td>
</tr>
<tr>
<td>&gt;&gt; x = 3;</td>
<td></td>
</tr>
<tr>
<td>%evaluation suppressed</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td></td>
</tr>
</tbody>
</table>

1.4.4 Comments:

<table>
<thead>
<tr>
<th>MATLAB</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>% a single line comment</td>
<td>// single line comment</td>
</tr>
<tr>
<td>v = 1584; %Initial velocity (km/h)</td>
<td>/* a multi-line</td>
</tr>
<tr>
<td></td>
<td>comment</td>
</tr>
<tr>
<td></td>
<td>*/</td>
</tr>
</tbody>
</table>

1.4.5 The clc command:

The clc command clears the Command Window. After a period of time working in the Command Window, the display may become long. The clc command puts it back to a blank window.

1.4.6 Running External Programs

You can run external programs from the MATLAB Command Window, i.e., an editor of your choice. The exclamation point character “!” is a shell escape and indicates that the rest of the
input line is a command to the operating system. This is useful for running other program without quitting MATLAB.

Example:

!emacs proj7.m

will invoke the emacs editor for the file called proj7.m. When you exit the external program, the operating system returns control to MATLAB.

1.4.7 Command History Window

The commands you enter in the Command Window are logged in the Command History window. In Command History, you can view previously used functions, and copy and paste to the Command Windows to execute your selected lines. You can erase your command history by selecting Clear Command History from the Edit menu.
1.4.8 Launch Pad Window

The Launch Pad gives easy access to the different tools within MATLAB. You may need to go through the Start Button (lower left-hand corner) to get to the Launch Pad.

1.4.9 Help

- Use the Help Browser to search for and view documentation. It is a web browser that displays HTML documents. There are many ways to get to the Help Browser:
  - click the help button ? in the toolbar
  - type helpbrowser in the Command Window
  - pick Help from the Launch Pad choices

1.4.10 Current Directory

MATLAB file operations use the current directory and the search path as reference points. Any file you want to run must either be in the current directory or on the search path. A quick way to view or to change the current directory is by using the Current Directory field on the desktop toolbar.

To search for, view, open, or make changes to MATLAB M-files related directories/files, use the Current Directory browser.

1.4.11 Editor/Debugger

The Editor Window is used for writing and editing programs and functions. This window is opened from the File option of the Main Menu Bar. The Editor/Debugger provides a graphical user interface for basic text editing along with M-file debugging. An example of an Editor Window is shown below.
You can display the contents of an M-file in the Command Window by using the `type` command.

>> type input_Example.m

%MatLab - Example of reading input from the keyboard
%check to make sure that the values are within range
%loop back if they are not

% all digits will be from 1 - 9
ones = [1 1 1 1];
nines = [9 9 9 9];

board = [ 0 0 0 0 ];
indices = [];

while length(indices) < 4
    guess = input('
Please enter your guesses--format of [a b c d] : ');
    %check if guesses are within range
    %if not within range, get another guess
    done = (guess==0);
if length(find(done)) == 4
    fprintf('Thanks for playing. I win - you gave up ')
    return
else
    inrange = (ones <= guess & guess <= nines);
    indices = find(inrange);
    if length(indices) < 4
        fprintf(' You need to enter values from 1 - 9 ');
    else
        fprintf(' Your code goes here

')
    end
end
fprintf ('Congratulations  You WIN !!!!!');