

César Pérez López

MATLAB Differential Equations

PRACTICAL HANDS-ON MATLAB SOLUTIONS

For your convenience Apress has placed some of the front matter material after the index. Please use the Bookmarks and Contents at a Glance links to access them.



Contents at a Glance

About the Author	ix
■ Chapter 1: Introducing MATLAB and the MATLAB Working Environment.....	1
■ Chapter 2: First Order Differential Equations. Exact Equations, Separation of Variables, Homogeneous and Linear Equations.....	33
■ Chapter 3: Higher Order Differential Equations. The Laplace Transform and Special Types of Equations	45
■ Chapter 4: Differential Equations Via Approximation Methods	61
■ Chapter 5: Systems of Differential Equations and Finite Difference Equations	67
■ Chapter 6: Numerical Calculus with MATLAB. Applications to Differential Equations.....	73
■ Chapter 7: Ordinary and Partial Differential Equations with Initial and Boundary Values.....	101
■ Chapter 8: Symbolic Differential and Integral Calculus.....	125



Introducing MATLAB and the MATLAB Working Environment

Introduction

MATLAB is a platform for scientific calculation and high-level programming which uses an interactive environment that allows you to conduct complex calculation tasks more efficiently than with traditional languages, such as C, C++ and FORTRAN. It is the one of the most popular platforms currently used in the sciences and engineering.

MATLAB is an interactive high-level technical computing environment for algorithm development, data visualization, data analysis and numerical analysis. MATLAB is suitable for solving problems involving technical calculations using optimized algorithms that are incorporated into easy to use commands.

It is possible to use MATLAB for a wide range of applications, including calculus, algebra, statistics, econometrics, quality control, time series, signal and image processing, communications, control system design, testing and measuring systems, financial modeling, computational biology, etc. The complementary toolsets, called *toolboxes* (collections of MATLAB functions for special purposes, which are available separately), extend the MATLAB environment, allowing you to solve special problems in different areas of application.

In addition, MATLAB contains a number of functions which allow you to document and share your work. It is possible to integrate MATLAB code with other languages and applications, and to distribute algorithms and applications that are developed using MATLAB.

The following are the most important features of MATLAB:

- It is a high-level language for technical calculation
- It offers a development environment for managing code, files and data
- It features interactive tools for exploration, design and iterative solving
- It supports mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
- It can produce high quality two-dimensional and three-dimensional graphics to aid data visualization
- It includes tools to create custom graphical user interfaces
- It can be integrated with external languages, such as C/C++, FORTRAN, Java, COM, and Microsoft Excel

The MATLAB development environment allows you to develop algorithms, analyze data, display data files and manage projects in interactive mode (see Figure 1-1).

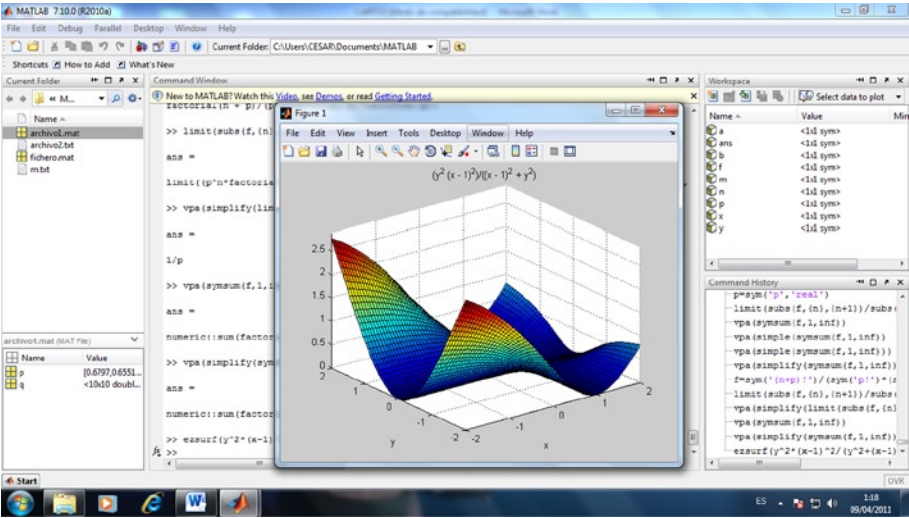


Figure 1-1.

Developing Algorithms and Applications

MATLAB provides a high-level programming language and development tools which enable you to quickly develop and analyze algorithms and applications.

The MATLAB language includes vector and matrix operations that are fundamental to solving scientific and engineering problems. This streamlines both development and execution.

With the MATLAB language, it is possible to program and develop algorithms faster than with traditional languages because it is no longer necessary to perform low level administrative tasks, such as declaring variables, specifying data types and allocating memory. In many cases, MATLAB eliminates the need for ‘for’ loops. As a result, a line of MATLAB code usually replaces several lines of C or C++ code.

At the same time, MATLAB offers all the features of traditional programming languages, including arithmetic operators, control flow, data structures, data types, object-oriented programming (OOP) and debugging.

Figure 1-2 shows a communication modulation algorithm that generates 1024 random bits, performs the modulation, adds complex Gaussian noise and graphically represents the result, all in just nine lines of MATLAB code.

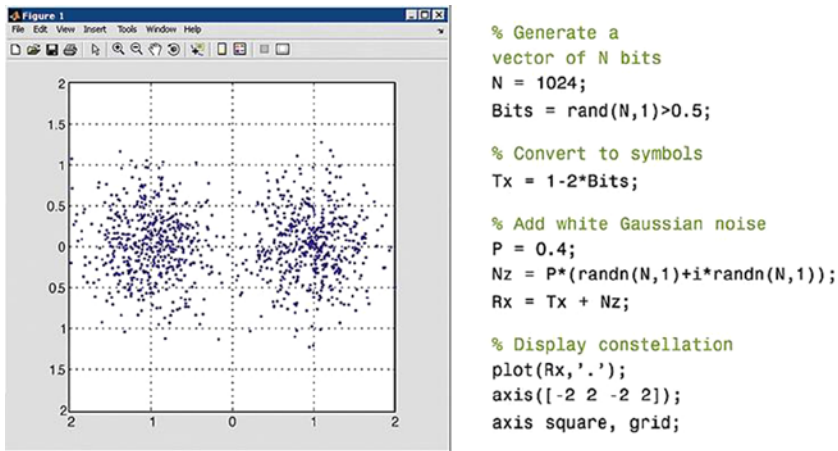


Figure 1-2.

MATLAB enables you to execute commands or groups of commands one at a time, without compiling or linking, and to repeat the execution to achieve the optimal solution.

To quickly execute complex vector and matrix calculations, MATLAB uses libraries optimized for the processor. For general scalar calculations, MATLAB generates instructions in machine code using JIT (*Just-In-Time*) technology. Thanks to this technology, which is available for most platforms, the execution speeds are much faster than for traditional programming languages.

MATLAB includes *development tools*, which help efficiently implement algorithms. Some of these tools are listed below:

- **MATLAB Editor** – used for editing functions and standard debugging, for example setting breakpoints and running step-by-step simulations
- **M-Lint Code Checker** - analyzes the code and recommends changes to improve performance and maintenance (see Figure 1-3)

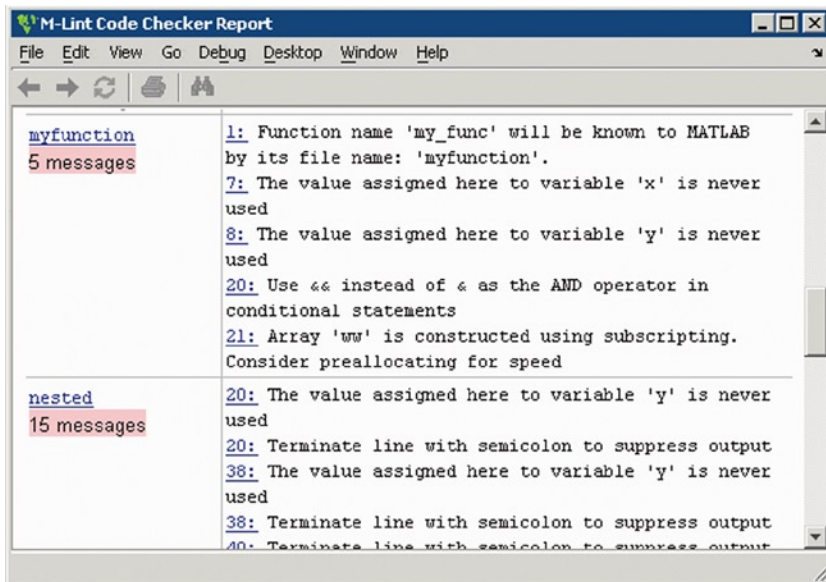


Figure 1-3.

- **MATLAB Profiler** - records the time taken to execute each line of code
- **Directory Reports** - scans all files in a directory and creates reports about the efficiency of the code, differences between files, dependencies of files and code coverage

You can also use the interactive tool GUIDE (*Graphical User Interface Development Environment*) to design and edit user interfaces. This tool allows you to include pick lists, drop-down menus, push buttons, radio buttons and sliders, as well as MATLAB diagrams and ActiveX controls. You can also create graphical user interfaces by means of programming using MATLAB functions.

Figure 1-4 shows a completed wavelet analysis tool (below) which has been created using the user interface GUIDE (above).

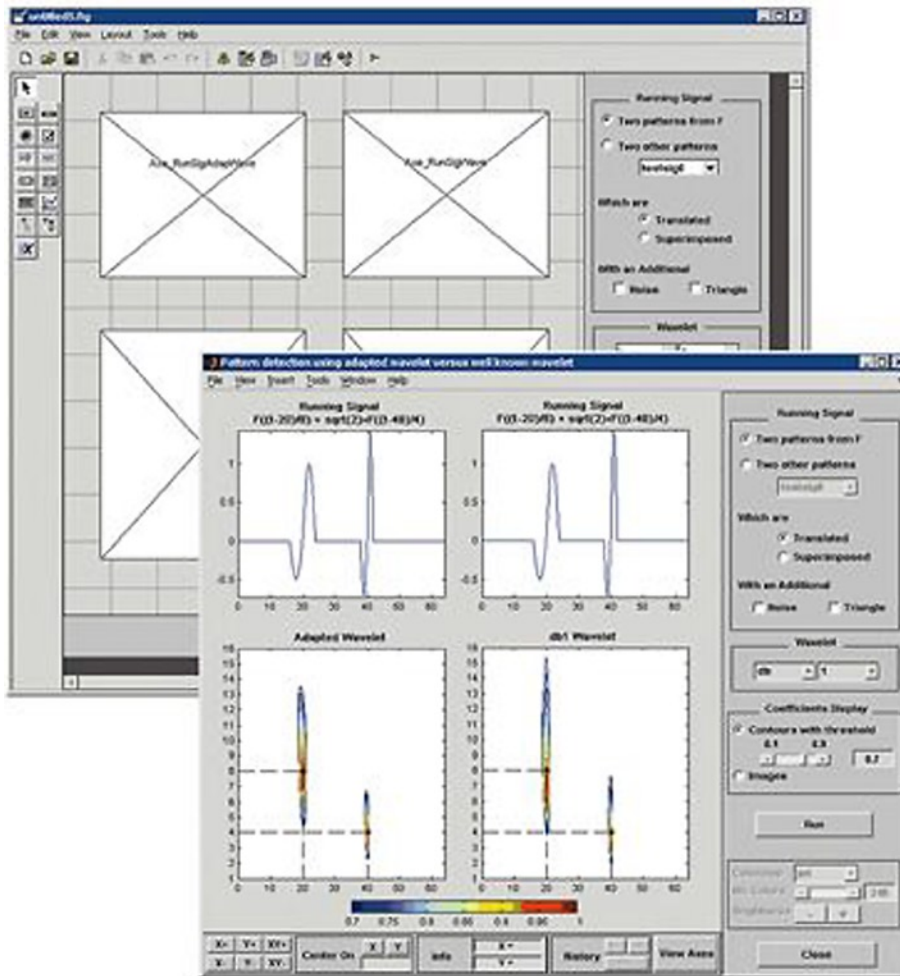


Figure 1-4.

Data Access and Analysis

MATLAB supports the entire process of data analysis, from the acquisition of data from external devices and databases, pre-processing, visualization and numerical analysis, up to the production of results in presentation quality.

MATLAB provides interactive tools and command line operations for data analysis, which include: sections of data, scaling and averaging, interpolation, thresholding and smoothing, correlation, Fourier analysis and filtering, searching for one-dimensional peaks and zeros, basic statistics and curve fitting, matrix analysis, etc.

The diagram in Figure 1-5 shows a curve that has been fitted to atmospheric pressure differences averaged between Easter Island and Darwin in Australia.

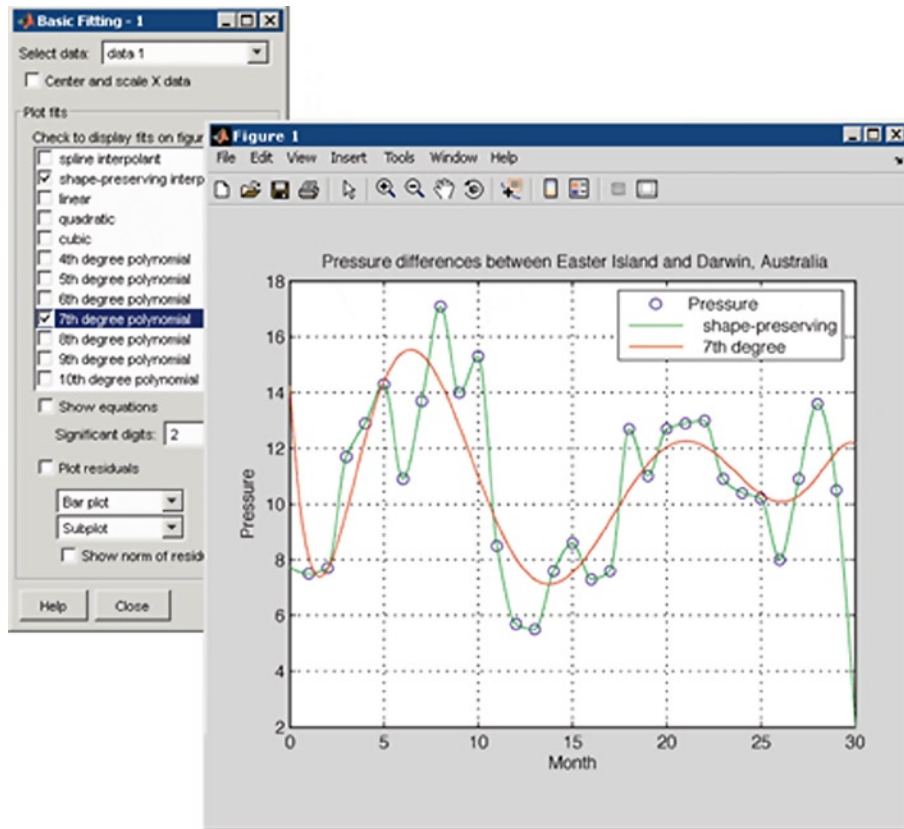


Figure 1-5.

The MATLAB platform allows efficient access to data files, other applications, databases and external devices. You can read data stored in most known formats, such as Microsoft Excel, ASCII text files or binary image, sound and video files, and scientific archives such as HDF and HDF5 files. The binary files for low level I/O functions allow you to work with data files in any format. Additional features allow you to view Web pages and XML data.

It is possible to call other applications and languages, such as C, C++, COM, DLLs, Java, FORTRAN, and Microsoft Excel objects, and access FTP sites and Web services. Using the Database Toolbox, you can even access ODBC/JDBC databases.

Data Visualization

All graphics functions necessary to visualize scientific and engineering data are available in MATLAB. This includes tools for two- and three-dimensional diagrams, three-dimensional volume visualization, tools to create diagrams interactively, and the ability to export using the most popular graphic formats. It is possible to customize diagrams, adding multiple axes, changing the colors of lines and markers, adding annotations, LaTeX equations and legends and plotting paths.

Various two-dimensional graphical representations of vector data can be created, including:

- Line, area, bar and sector diagrams
- Direction and velocity diagrams

- Histograms
- Polygons and surfaces
- Dispersion bubble diagrams
- Animations

Figure 1-6 shows linear plots of the results of several emission tests of a motor, with a curve fitted to the data.

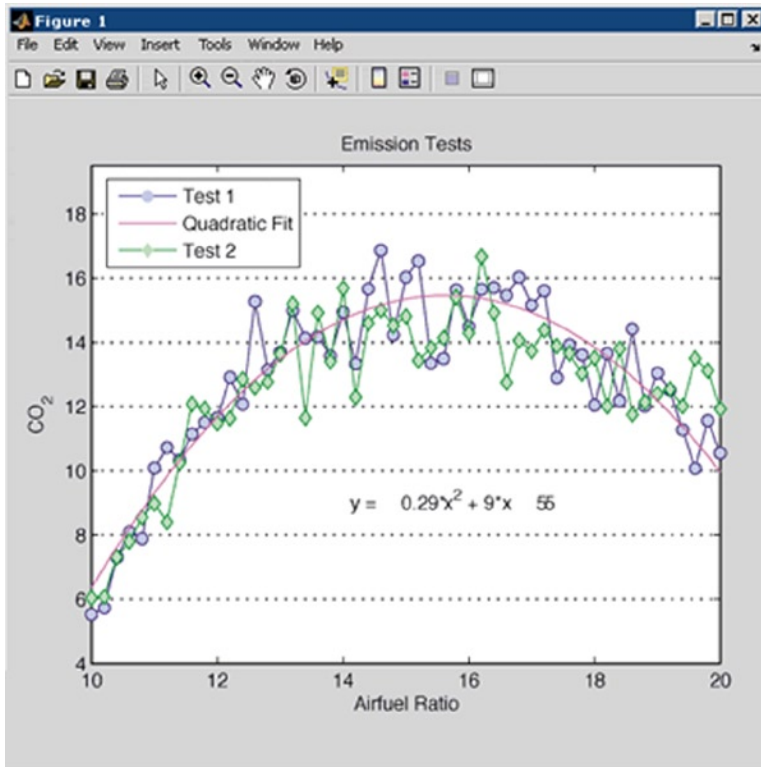


Figure 1-6.

MATLAB also provides functions for displaying two-dimensional arrays, three-dimensional scalar data and three-dimensional vector data. It is possible to use these functions to visualize and understand large amounts of complex multi-dimensional data. It is also possible to define the characteristics of the diagrams, such as the orientation of the camera, perspective, lighting, light source and transparency. Three-dimensional diagramming features include:

- Surface, contour and mesh plots
- Space curves
- Cone, phase, flow and isosurface diagrams

Figure 1-7 shows a three-dimensional diagram of an isosurface that reveals the geodesic structure of a fullerene carbon-60 molecule.

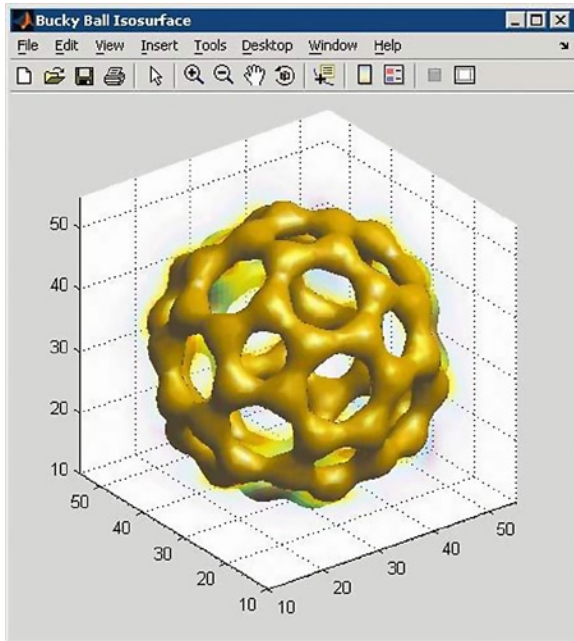


Figure 1-7.

MATLAB includes interactive tools for graphic editing and design. From a MATLAB diagram, you can perform any of the following tasks:

- Drag and drop new sets of data into the figure
- Change the properties of any object in the figure
- Change the zoom, rotation, view (i.e. panoramic), camera angle and lighting
- Add data labels and annotations
- Draw shapes
- Generate an M-file for reuse with different data

Figure 1-8 shows a collection of graphics which have been created interactively by dragging data sets onto the diagram window, making new subdiagrams, changing properties such as colors and fonts, and adding annotations.

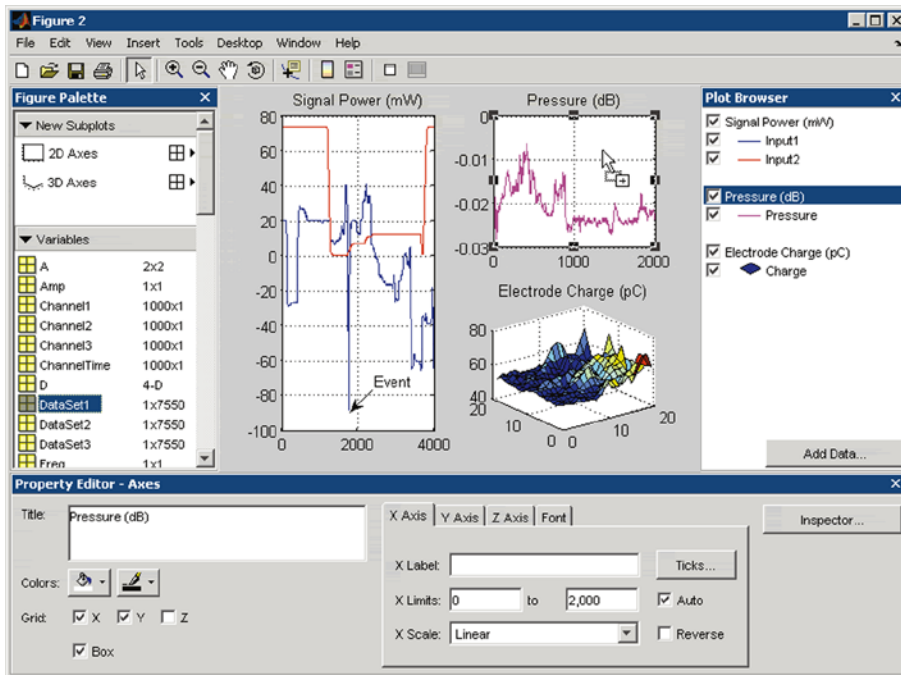


Figure 1-8.

MATLAB is compatible with all the well-known data file and graphics formats, such as GIF, JPEG, BMP, EPS, TIFF, PNG, HDF, AVI, and PCX. As a result, it is possible to export MATLAB diagrams to other applications, such as Microsoft Word and Microsoft Powerpoint, or desktop publishing software. Before exporting, you can create and apply style templates that contain all the design details, fonts, line thickness, etc., necessary to comply with the publication specifications.

Numerical Calculation

MATLAB contains mathematical, statistical, and engineering functions that support most of the operations carried out in those fields. These functions, developed by math experts, are the foundation of the MATLAB language. To cite some examples, MATLAB implements mathematical functions and data analysis in the following areas:

- Manipulation of matrices and linear algebra
- Polynomials and interpolation
- Fourier analysis and filters
- Statistics and data analysis
- Optimization and numerical integration
- Ordinary differential equations (ODEs)
- Partial differential equations (PDEs)
- Sparse matrix operations

Publication of Results and Distribution of Applications

In addition, MATLAB contains a number of functions which allow you to document and share your work. You can integrate your MATLAB code with other languages and applications, and distribute your algorithms and MATLAB applications as autonomous programs or software modules.

MATLAB allows you to export the results in the form of a diagram or as a complete report. You can export diagrams to all popular graphics formats and then import them into other packages such as Microsoft Word or Microsoft PowerPoint. Using the MATLAB Editor, you can automatically publish your MATLAB code in HTML format, Word, LaTeX, etc. For example, Figure 1-9 shows an M-file (left) published in HTML (right) using the MATLAB Editor. The results, which are sent to the Command Window or to diagrams, are captured and included in the document and the comments become titles and text in HTML.

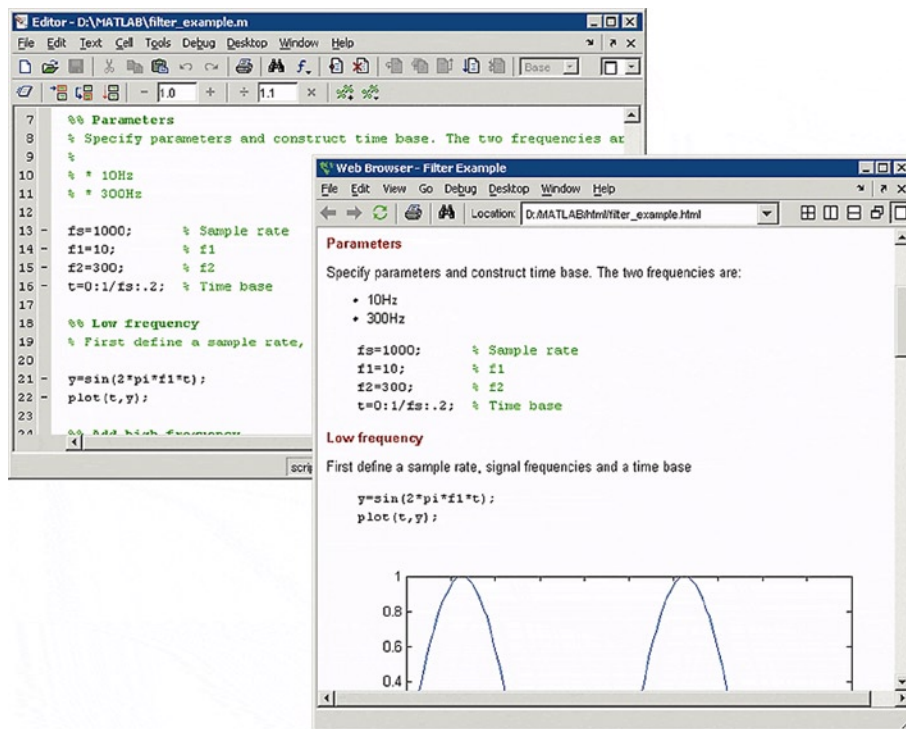


Figure 1-9.

It is possible to create more complex reports, such as mock executions and various parameter tests, using MATLAB Report Generator (available separately).

MATLAB provides functions enabling you to integrate your MATLAB applications with C and C++ code, FORTRAN code, COM objects, and Java code. You can call DLLs and Java classes and ActiveX controls. Using the MATLAB engine library, you can also call MATLAB from C, C++, or FORTRAN code.

You can create algorithms in MATLAB and distribute them to other users of MATLAB. Using the MATLAB Compiler (available separately), algorithms can be distributed, either as standalone applications or as software modules included in a project, to users who do not have MATLAB. Additional products are able to turn algorithms into a software module that can be called from COM or Microsoft Excel.

The MATLAB working environment

Figure 1-10 shows the primary workspace of the MATLAB environment. This is the screen in which you enter your MATLAB programs.

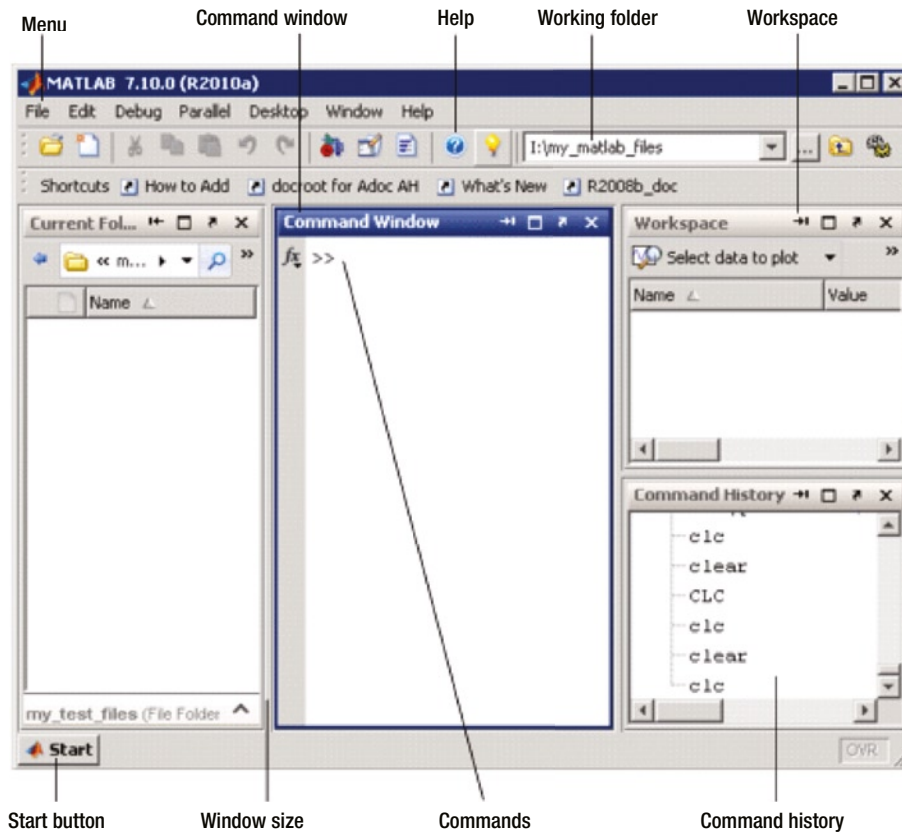


Figure 1-10.

The following table summarizes the components of the MATLAB environment.

Tool	Description
Command History	This allows you to see the commands entered during the session in the Command Window, as well as copy them and run them (lower right part of Figure 1-11)
Command Window	This is where you enter MATLAB commands (central part of Figure 1-11)
Workspace	This allows you to view the contents of the workspace (variables, etc.) (upper right part of Figure 1-11)
Help	This offers help and demos on MATLAB
Start button	This enables you to run tools and provides access to MATLAB documentation (Figure 1-12)

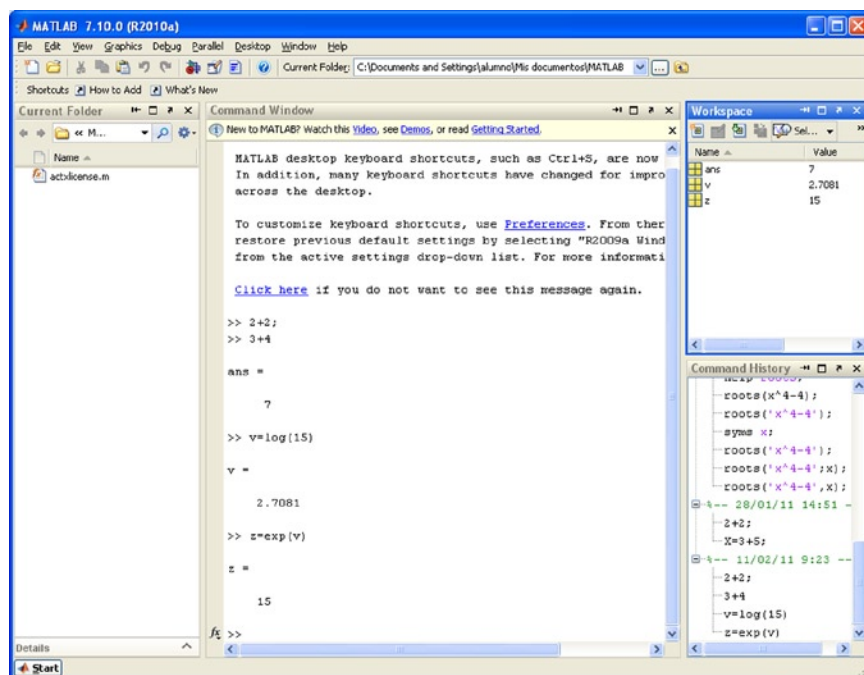


Figure 1-11.

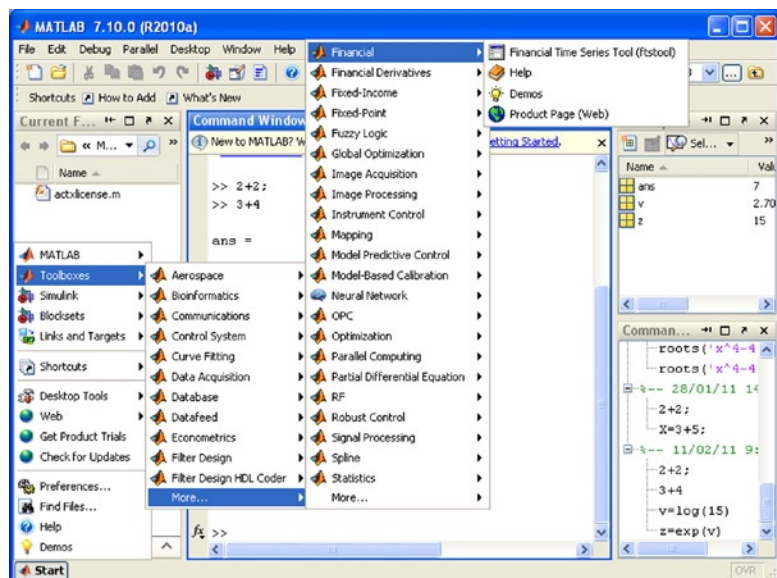


Figure 1-12.

MATLAB commands are written in the Command Window to the right of the user input prompt “`>>`” and the response to the command will appear in the lines immediately below. After exiting from the response, the user input prompt will re-display, allowing you to input more entries (Figure 1-13).

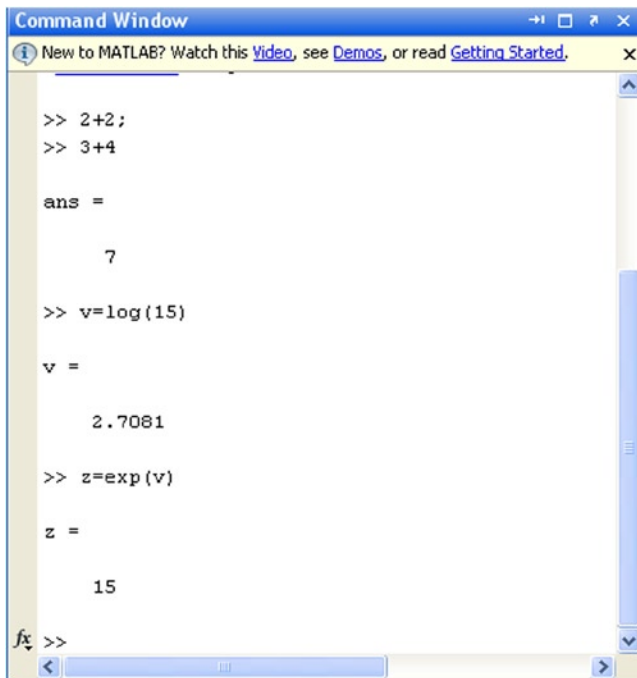


Figure 1-13.

When an input is given to MATLAB in the Command Window and the result is not assigned to a variable, the response returned will begin with the expression “*ans=*”, as shown near the top of Figure 1-13. If the results are assigned to a variable, we can then use that variable as an argument for subsequent input. This is the case for the variable *v* in Figure 1-13, which is subsequently used as the input for an exponential.

To run a MATLAB command, simply type the command and press *Enter*. If at the end of the input we put a semicolon, the program runs the calculation and keeps it in memory (*Workspace*), but does not display the result on the screen (see the first entry in Figure 1-13). The input prompt “`>>`” appears to indicate that you can enter a new command.

Like the C programming language, MATLAB is case sensitive; for example, $\text{Sin}(x)$ is not the same as $\text{sin}(x)$. The names of all built-in functions begin with a lowercase character. There should be no spaces in the names of commands, variables or functions. In other cases, spaces are ignored, and they can be used to make the input more readable. Multiple entries can be entered in the same command line by separating them with commas, pressing *Enter* at the end of the last entry (see Figure 1-14). If you use a semicolon at the end of one of the entries in the line, its corresponding output will not be displayed.

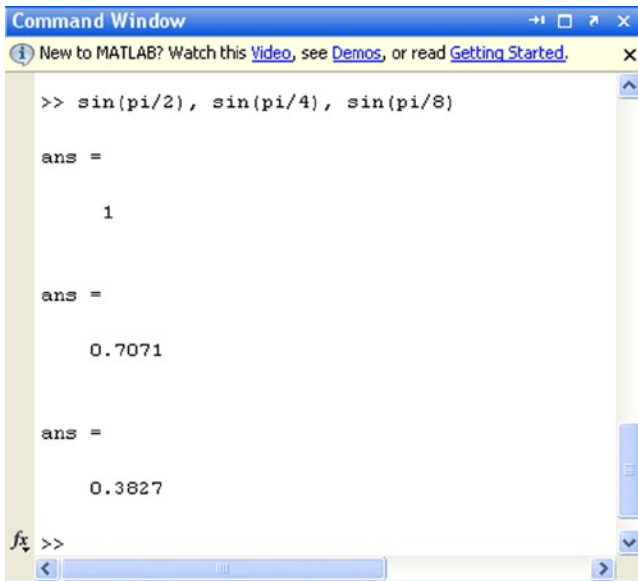


Figure 1-14.

Descriptive comments can be entered in a command input line by starting them with the “%” symbol. When you run the input, MATLAB ignores the comment and processes the rest of the code (see Figure 1-15).

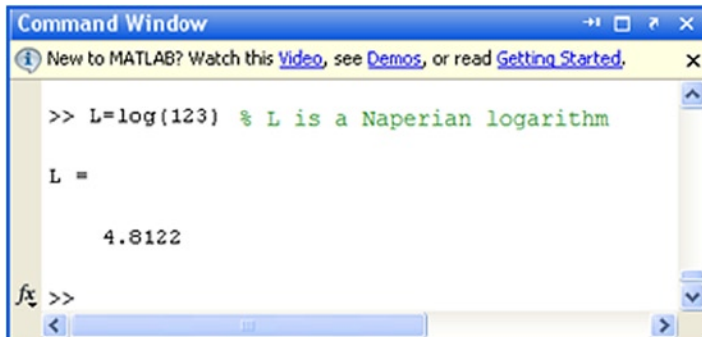


Figure 1-15.

To simplify the process of entering script to be evaluated by the MATLAB interpreter (via the Command Window prompt), you can use the arrow keys. For example, if you press the up arrow key once, you will recover the last entry you submitted. If you press the up key twice, you will recover the penultimate entry you submitted, and so on.

If you type a sequence of characters in the input area and then press the up arrow key, you will recover the last entry you submitted that begins with the specified string.


Commands entered during a MATLAB session are temporarily stored in the buffer (*Workspace*) until you end the session, at which time they can be permanently stored in a file or are permanently lost.

Below is a summary of the keys that can be used in MATLAB's input area (command line), together with their functions:

Up arrow (Ctrl-P)	Retrieves the previous entry.
Down arrow (Ctrl-N)	Retrieves the following entry.
Left arrow (Ctrl-B)	Moves the cursor one character to the left.
Right arrow (Ctrl-F)	Moves the cursor one character to the right.
CTRL-left arrow	Moves the cursor one word to the left.
CTRL-right arrow	Moves the cursor one word to the right.
Home (Ctrl-A)	Moves the cursor to the beginning of the line.
End (Ctrl-E)	Moves the cursor to the end of the current line.
Escape	Clears the command line.
Delete (Ctrl-D)	Deletes the character indicated by the cursor.
Backspace	Deletes the character to the left of the cursor.
CTRL-K	Deletes (kills) the current line.

The command *clc* clears the command window, but does not delete the contents of the work area (the contents remain in the memory).

Help in MATLAB

You can find help for MATLAB via the help button  in the tool bar or via the *Help* option in the menu bar. In addition, support can also be obtained via MATLAB commands. The command *help* provides general help on all MATLAB commands (see Figure 1-16). By clicking on any of them, you can get more specific help. For example, if you click on the command *graph2d*, you get support for two-dimensional graphics (see Figure 1-17).

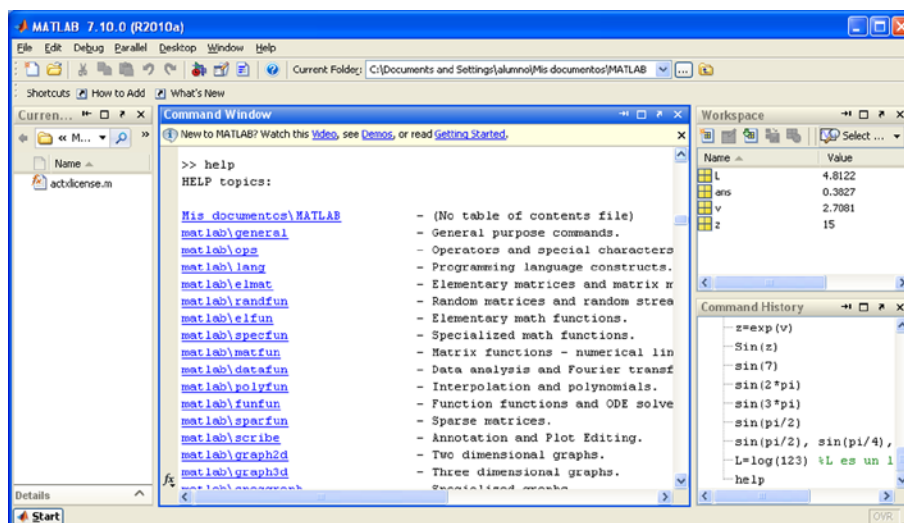


Figure 1-16.