CAEN TECHNICAL NOTE
UNIVERSITY OF MICHIGAN • COLLEGE OF ENGINEERING • COMPUTER AIDED ENGINEERING NETWORK

# Introduction to MATLAB

## Introduction

MATLAB is an interactive package for numerical analysis, matrix computation, control system design, and linear system analysis and design available on most CAEN platforms (Mac, Windows, Solaris, and Linux). In addition to the standard functions provided by MATLAB, there are thirteen toolboxes, or collections of functions and procedures, available as part of the MATLAB package. The toolboxes are:

- Control System: Provides several features for advanced control system design and analysis.
- Communications: Provides functions to model the components of a communication system's physical layer.
- Signal Processing: Contains functions to design analog and digital filters and apply these filters to data and analyze the results.
- System Identification: Provides features to build mathematical models of dynamical systems based on observed system data.
- Robust Control: Allows users to create robust multivariable feedback control system designs based on the concept of the singular-value Bode plot.
- Simulink: Allows you to model dynamic systems graphically.
- Neural Network: Allows you to simulate neural networks.
- Fuzzy Logic: Allows for manipulation of fuzzy systems and membership functions.
- Image Processing: Provides access to a wide variety of functions for reading, writing, and filtering images of various kinds in different ways.
- Analysis: Includes a wide variety of system analysis tools for varying matrices.
- Optimization: Contains basic tools for use in constrained and unconstrained minimization problems.
- Spline: Can be used to find approximate functional representations of data sets.
- Symbolic: Allows for symbolic (rather than purely numeric) manipulation of functions.
- User Interface Utilities: Includes tools for creating dialog boxes, menu utilities, and other user interaction for script files.

## Documentation

Each toolbox has its own manual that is available for checkout from the Media Union Reserve Desk along with the MATLAB manual. In addition to manuals specific to the toolboxes listed above, the following manuals are also available at the Media Union Reserves Desk:

- *MATLAB Building a Graphical User Interface*

- *MATLAB External Interface Guide for UNIX Workstations*
- *MATLAB Reference Guide*
- *MATLAB Release Notes*
- *MATLAB State-Space Identification Toolbox*
- *MATLAB User's Guide for UNIX Workstations*

## Getting Started

### On a PC

The MATLAB program is found under the Start button. Then click **Programs > Math and Numerical Methods > MATLAB**. Printing on the PC is done by selecting Print from the File menu at the top of the window. Printing from the command window will print the command window text, while a print from the graph window will print the graph.

### On a UNIX/Linux Workstation

To start MATLAB on any UNIX/Linux workstation, type **matlab**. If you receive a **Permission Denied** error, use the **klog -c engin.umich.edu** command to authenticate. To terminate MATLAB, type **quit** at the **>>** prompt. To set a specific printer type **setenv PRINTER *printername*** at the UNIX shell prompt before starting MATLAB and set the PRINTER variable to the desired printer. Once in the MATLAB program, typing **print** will print the most recent graph displayed.

### On a Mac

You can find MATLAB in the Applications folder. After opening the program, the MATLAB Command window and an accompanying Graph window will appear. The Command window allows users to enter commands at the **>>** prompt and displays numerical output. The output of any plotting command is displayed in the Graph window. To exit MATLAB, select Quit from the File menu.

There are two ways to obtain output from the Mac. The first way is to select Print from the File menu. If you are in a Command or Edit window, text is sent to the printer. If a Graph window is active, the graph is printed. Alternatively, you can enter the **prtsc** command in the Command window to print a graph.

## Online Help

During any MATLAB session, online help is available for a variety of topics. To see a list of help topics, type **help** in the Command window. MATLAB is case sensitive, so all commands and variable names must be entered in lower case. For help on a specific topic, type **help *topic*** (e.g. **help sum**).

On the Mac and PC, you also can get help from the menu bar. Select **About MATLAB** from the Apple menu and choose the desired category. Click on the Help button. Scroll to the desired topic, select it with the mouse, and click on the Help button. After you are finished reading the Help entry, click on Topics to move back up to the list of Help items. Then you can choose another item and click on the Help button again or you can click on Topics to return you to the list of Help Categories. Clicking the close box in the upper left corner closes the Help window. Mathworks also offers help on the web at:

**http://www.mathworks.com/products/matlab/**

## Session Transcripts and Permanent Variables

The **diary** command can be used to create a transcript of a MATLAB session. Once a diary command is issued, all subsequent commands and output are written to the transcript file. To create a transcript file called session1, for example, type **diary session1**.

The **who** command displays the names of all currently defined variables. MATLAB also has several permanent variables defined for frequently used numbers that appear in the display of currently defined variables. To display the value of a variable, just type the variable name. For example, type **pi** to display the value of pi.

The **format** command can be used to change the way variables are printed. Short format, the default, displays four decimal places while long format displays fourteen. Type **format long** to change the display format.

Close the transcript file using the **diary off** command at the end of the session.

### Defining and Clearing Variables

The basic data element in MATLAB is the matrix. User defined variables are assigned values by entering a single value (scalar), a one-dimensional table (vector), or a two-dimensional table (matrix). Variables are defined by typing the name of the variable followed by an equal sign and the value of the variable. Brackets enclose the values of a vector or matrix and semicolons separate rows of matrices. Some examples are given here:

| | |
|---|---|
| **a = 1** | (lets a equal a scalar) |
| **b = [ 1 2 3 ]** | (lets b equal a vector) |
| **c = [ 1 2 3 ; 4 5 6 ; 7 8 9 ]** | (lets c equal a matrix) |

Notice that the value of the variable is displayed after each variable is assigned. To suppress the output from any command, follow the command with a semicolon. To assign a new value to a variable, type the name followed by an equal sign and the new value.

| | |
|---|---|
| **a** | (displays the value of a) |
| **a = 2;** | (changes a but does not display the value) |

Sometimes the stack may become cluttered with old variables and temporary results. The **clear** command clears all user-defined variables. It can be used to clear all variables or it may be followed by the name of a variable, in which case it clears only the specified variable. For example:

| | |
|---|---|
| **who** | (lists the currently defined variables) |
| **clear a** | (clears a) |
| **clear** | (clears all user defined variables) |

## Matrix Entry Shortcuts

MATLAB provides several commands to create commonly used matrices.

| | |
|---|---|
| **a = eye(3)** | (creates a 3x3 identity matrix) |
| **b = ones(3)** | (creates a 3x3 matrix of ones) |
| **c = diag( [ 1 2 3 ] )** | (creates a matrix whose diagonal is 1 2 3) |

Matrices with regularly increasing or decreasing values can be created as follows:

| | |
|---|---|
| **d = [ 1:5 ]** | (creates a vector with entries 1 2 3 4 5) |
| **e = [ 1:3 ; 1:3 ; 1:3 ]** | (creates a 3x3 matrix) |
| **f = [ 1:0.1:2 ]** | (creates a vector with entries from 1 to 2 incremented by 0.1) |

## Displaying Data

When it is convenient to examine a specific element, row, or column of a matrix rather than the entire matrix, a single element of a matrix can be addressed by typing the name of the matrix followed by the row and column indices of the element in parentheses. To address an entire column or row, a colon can be substituted for either index. In this case, the colon can be thought of as a wildcard character. For example:

| | |
|---|---|
| **clear** | (clears all user defined variables) |
| **a = [ 1 2 3 ; 4 5 6 ; 7 8 9 ]** | (defines the matrix *a*) |
| **a(1,1)** | (displays the element in the first row, first column of *a*) |
| **a(:,1)** | (displays first column of *a*) |

A partition of a matrix is addressed by inputting a vector rather than a scalar as either index.

| | |
|---|---|
| **a( [ 1 2 ] , [ 1 2 ] )** | (displays the first two rows and columns of *a*) |
| **a( [ 2 3 ] , : )** | (displays the second and third rows of *a*) |

## Matrix Arithmetic and Functions

The conjugate transpose of a matrix can be found by placing an apostrophe after the name of the matrix. (If the matrix is purely real, then this will be the same as the transpose. Otherwise, a period will need to precede the apostrophe in the command.) If the matrix *a* is not already defined from the last section, it will have to be defined.

| | |
|---|---|
| **a** | (displays the matrix *a*) |
| **a'** | (displays the conjugate transpose of *a*) |
| **a.'** | (displays the transpose of *a*) |

Matrices can be added to each other, subtracted from one another, and multiplied with other matrices or vectors, as long as they have the proper dimensions:

| | |
|---|---|
| **i = eye(3)** | (defines the identity matrix *i*) |
| **a + a'** | (displays *a* + *a* conjugate transpose) |
| **a * i** | (displays *a* * *i*) |

Division of matrices by vectors or other matrices is not defined. That is, it is impossible to divide by a matrix or vector. Instead, the program multiplies by the inverse of a matrix. In MATLAB there are two types of division denoted by / and \. These two operations do not represent true division, but rather shortcuts for multiplication by the inverse of a matrix.

If A is a square matrix, then A \ B is equivalent to INV(A)*B and gives the solution to the set of linear equations A*X = B. Similarly, B / A is equivalent to B*INV(A) and gives the solution to X*A = B. For example:

| | |
|---|---|
| **a = [ 2 1 ; 1 2 ];** | (defines *a*) |
| **b = [ 1 -3 ; 3 9 ];** | (defines *b*) |
| **a\b** | (equivalent to inv(a)*b) |
| **b/a** | (equivalent to b*inv(a)) |

MATLAB is able to calculate simple functions of matrices very easily. Some examples are given below:

| | |
|---|---|
| **det(a)** | (calculates the determinant of *a*) |
| **inv(a)** | (calculates the inverse of *a*) |
| **rank(a)** | (calculates the rank of *a*) |

## Conditional and Looping Commands

### for Loops

The **for** command can be used to execute a command or a sequence of commands for a specified number of iterations. The format of the **for** command is similar to the format of the BASIC *for* statement or the FORTRAN *do* statement. The **for** command can be entered on one or several lines to allow several statements to be executed within the loop. The following example creates an array containing the squares of the numbers one through ten.

| | |
|---|---|
| **for i = 1:10,** | (initializes loop) |
| **a(i) = i^2;** | (assigns values for the array *a*) |
| **end;** | (ends loop) |
| **a** | (displays *a*) |

Notice that the method used for specifying the range for *i* is the same as the shortcut discussed earlier for creating matrices. The beginning and ending values for *i* are separated by a colon.

### while Loops

The **while** statement allows conditional looping. The command has the same general form as the **for** command except that the variable and range are replaced by a relational operator and two operands.

| | |
|---|---|
| **i = 0;** | (initializes *i*) |
| **while i < 10,** | (begins loop) |
| **i = i + 1;** | (increments *i*) |
| **a(i) = i^2;** | (assigns values for the array *a*) |
| **end;** | (ends loop) |

```
a                       (displays a)
```

This has the same effect as the first example of the **for** statement. The valid relational operators are == (equal), >, <, <=, >=, and ~= (not equal). The two operands must be scalars or expressions that result in scalars.

## if, else, and elseif Statements

The **if** command provides for the conditional execution of a statement. The following example creates an identity matrix and illustrates nested for statements as well as the **if** statement.

```
for i = 1:3,            (initializes i)
for j = 1:3,            (initializes j)
if i == j a(i,j)=1;     (put ones on diagonal)
else a(i,j)=0;          (and zeros everywhere else)
end;                    (ends if statement)
end;                    (ends j loop)
end;                    (ends i loop)
```

When MATLAB is first invoked, the variables *i* and *j* are pre-assigned to be the imaginary square root of -1. However, once *i* or *j* is assigned to another value (e.g. as a loop counter), it will retain this value throughout the MATLAB session. Thus, if work with complex numbers inside MATLAB is desired, loop counters other than *i* or *j* should be used. To reset *i* to its default value, you can either type **clear i** or **i=sqrt(-1)**. Alternatively, any references to the imaginary number *i* in scripts can be modified to refer to sqrt(-1) to be absolutely safe.

## Exiting MATLAB

Remember to type **diary off** if you created a transcript session. You then will have a complete transcript of your MATLAB session. To leave MATLAB, type **exit** or **quit** at the **>>** prompt or select File -> Quit on a PC or Mac.