Guide to Matlab Programming (Version 1.2)
Prepared by Dr. C. S. Tritt

About This Document and Getting Help

This document is intended to be used with Matlab's extensive built in and online help system. It is not so much a quick reference as a collection of terms and concepts that is intended to make it easier to look up help information. Access the Matlab Help Browser window from Matlab Desktop using *Help* | *Matlab Help* or by pressing *F1* while in the Command Window. The command `help /` lists all Matlab operators and Special Characters (`[]`, `()`, `{}`, `=`, `'`, `.`, `...`, `,`, `;`, `:`, `%`, `!` and `@`). Corrections and suggestions to tritt@msoe.edu are encouraged.

Matlab Desktop

Important Windows include: Command Window – Matlab's primary user interface, activated from Desktop using *Desktop* | *Command Window*; Current Directory Browser – Used to set default folders, activated using *Desktop* | *Current Directory*; Editor/Debugger – Used to edit programs, etc., activated using *File* | *New* | *M-file* or by entering `edit` *`filename`* in the Command Window (always remember to save your work before attempting to execute your script); Workspace Browser – Used to view variable values, etc., activated using *Desktop* | *Workspace*.

Matlab Path and Current Directory

When a token is entered in the Command Window or specified an m-file, Matlab attempts to resolve it in a specified way. It first searches the current Workspace for a variable having the specified name. It then searches the "Current Directory" for an m-file having the specified name. The Current Directory can and should be specified in the Matlab Desktop. Finally, it searches the Matlab Search Path for an m-file having the specified name. The Search Path can be managed in the Desktop using *File* | *Set Path* or the Command Window using `path`, `path2rc`, `addpath` or `rmpath` commands. The `which` command can be used to determine which m-file is currently being found via the search process.

M-files, M-file Comments and Output Suppression

M-files provide a relatively simple but quite powerful scripting interface to the Matlab environment. M-files contain a sequence of Matlab commands that are executed (interpreted) by Matlab when the file is "run." M-files can also contain control structures that dictate the sequential, selective or repetitive execution of individual commands within the file. Comment lines in m-files are denoted by beginning them with the `%` symbol. The normal Command Window (console) output generated by a command may be suppressed by ending a m-file line with a `;` symbol. Proper use of comments is extremely important in Matlab as it's built in help system uses these comments and much Matlab programming is done by multiple individuals over extended time periods. **M-file names should not contain spaces.**

Program Script Structure

Matlab program m-files should always begin with the following comment lines: script and file name, purpose, inputs (including units), output (including units), revision history (including programmer names and revision dates) and a list of variables used. I recommend a `clear` and `clc` statements be used to clear the workspace and command window at the start of most programs. Simple programs can be created using the `input` and `fprintf` functions for console input and output or `inputdlg`, `menu` and `msgbox` functions for GUI input and output. The first block of comments in a program file is displayed when "`help` *`filename`*" is entered in the Command Window. See my tconv.m and guitconv.m files for examples.

Function Script Structure

Functions are Matlab scripts intended to be called by other scripts. Function scripts always start with a function statement. The function statement consists of the word function, the name of the variable returned by the function, an = sign, the function name (which must match the m-file name) and the functions input arguments enclosed in parenthesis and separated by comas. Next should come a block of comments. The first comment line will be searched and displayed by `lookfor` command and should be a concise description of the purpose of the function. The remaining comments up to the first blank line will be displayed by the `help` *`functionname`* command and should specify the purpose of the function, its calling sequence, its input arguments (including data types and units), its outputs (including data type and units) and its revision history. Next comes the actually Matlab commands that make up the function. The function must end with an `end` statement and may include one or more `return` statements. See my f2k.m function for an example.

Variable Names and Data Types

Matlab variable names are **case sensitive**. Names must begin with a letter. They may contain letters, digits and underscores. **Variable names should not contain spaces.** Only the first 63 characters are significant. The name `pi` is predefined as 3.1415…. Be very careful about using built-in function names and special Matlab values (like `i` and `true`) as variable names as the variable will block access to the function or special value.

Matlab is the only modern computer language that is not strongly typed. Variables need not be declared prior to use. All variables are specialized types of arrays. Their size can range from 0 by 0 (empty) to n-dimensional. They may contain `logical`, `char`, Numeric, `cell`, `structure`, `function handle` or other contents. Numeric types include a variety of integer (like `int8` and `uint64`) and floating point (like `single` and `double`) types. To create an non-double type numeric variable use the desired type function in the assignment as in `myInt = int8(23)`. The default numerical data type is `double` that can contain purely real, imaginary or complex values. The type double can contain values between 1.79769e+308 to -2.22507e-308 and 2.22507e-308 to 1.79769e+308. They can also contain positive and negative infinity (`Inf` and `–Inf`, respectively) and not a number (`NaN`).

Matlab effective passes function arguments by value, but actually passes by reference unless the receiving function attempts to modify them.

Character Strings

In Matlab, text is stored in 1-dimensional character arrays called strings. String literals are denoted using single quotes as in, `'This is a literal'`. Some useful string functions include (see Table 6.3 in Chapman for more information): `strcat` (see also the concatenation operator `[]`), `strcmp`, `findstr`, `int2str`, `num2str`, `sprintf`, `eval`, `str2double` and `sscanf`. The equality operator (==) can only be used to compare strings of equal length. The relational operators (like <, >, <= and >=) can't be used to compare strings.

Cell Arrays

Cell arrays are arrays of pointers that can effectively hold multiple types of data. Cell arrays are particularly useful for holding collections of strings (because unlike 2-D character arrays, the strings can have different sizes) and passing data of multiple types to and from functions.

Structures

Structures are collections of data values, possibly of multiple types, accessible by field name. Arrays of structures are possible. Arrays of structures can be accessed in vectorized (unsubscripted) fashion and structures can be accessed as a whole by omitting field names.

Handle Graphics

Handle graphics refers to the use of "handles" to graphical items that are effectively structures the content of which control the appearance of the corresponding item. The content of handles is thought of as collection of *property*, *value* pairs. Properties can be changed using the GUI based Property Editor or programmatically. When hold is on, each plot statement appears to return a handle to a different line on the composite plot.