CS 412
UW-Madison

MATLAB Reference Sheet

created by
Giordano Fusco,
version 01-22-06

Useful workspace functions (`help general`)

`help`	obtain help generally or for a specific function
`lookfor`	obtain one-line help if it exists
`more`	toggles pagination, useful for longs "helps"
`load`	read variables from a file
`save`	save all or selected variables to a file

Path and environment (`help general`)

`cd`	change to a specific directory
`dir`	list files in the current directory
`path`	display or modify the function search path

Basic types

The basic variable type is a two-dimensional array of doubles (64-bit representation).
A *scalar* is a 1×1 array.
A *row vector* of length n is a $1 \times n$ array.
A *column vector* of length m is an $m \times 1$ array.
A *matrix* of dimensions m rows and n columns is an $m \times n$ array.

Variable name conventions

MATLAB is case sensitive.
A variable must start with a letter (A-Z, a-z).
Up to 31 letters, digits and underscores.

Default variables/constants (`help elmat`)

`ans`	result of last unassigned calculation
`eps`	smallest number that can be added to 1.0 and still be different
`flops`	count of floating point operations
`Inf`	infinity, e.g. 1/0 = `Inf`
`NaN`	Not a Number, e.g. 0/0 = `NaN`
`pi`	value of π (3.1415...)
`i, j`	$\sqrt{-1}$
`realmax`	largest real number MATLAB can represent
`realmin`	smallest real number MATLAB can represent

About user variables (`help elmat,help general`)

`clear`	clear all or selected variables (or functions) from the current workspace
`length`	length of a vector or maximum dimension of an array
`size`	display dimensions of a particular array
`who`	display current workspace variable names
`whos`	display current workspace variable names, types and associated sizes

Formatting (`help format`)

`format short`	scaled fixed point format with 5 digits
`format long`	scaled fixed point format with 15 digits
`format compact`	suppress extra line-feeds
`format loose`	put extra line-feeds in the output

Punctuation (`help punct`)

`.`	decimal point, e.g. 325/100, 3.25 and .325e1 are equivalent
`...`	three or more decimal points at the end of a line cause the following line to be a continuation
`,`	comma is used to separate matrix elements and arguments to functions, also used to separate statements in multi-statement lines
`;`	used inside brackets to indicate the ends of the rows of a matrix, also used after an expression or statement to suppress printing
`%`	begins comments
`'`	quote. `'ANY TEXT'` is a vector whose components are the ASCII codes for the characters. A quote within the text is indicated by two quotes, e.g. `'Don''t forget.'`

Explicit matrix creation

Elements in a row can be delimited by a comma or a space.
Explicit assignment using ;'s to end rows

```
  a = [1,2,3;4,5,6;7,8,9]
```

Explicit assignment using "newline" to end rows

```
  a = [1,2,3
  4,5,6
  7,8,9]
```

Explicit assignment using continuation lines

```
  b = [1 2 3 4 5 6 ...
  7 8 9 10]
```

Vector/Matrix initialization (`help elmat`)

`linspace(a,b,N)`	linearly spaced intervals between a and b (inclusive) comprised of N points
`zeros(m,n)`	an m by n array of zeroes
`zeros(n)`	an n by n array of zeroes
`ones(m,n)`	an m by n array of ones
`ones(n)`	an n by n array of ones
`eye(m,n)`	an m by n array with ones on the diagonal
`eye(n)`	an n by n identity matrix
`ones(n)`	an n by n array of ones
`rand(m,n)`	an m by n array of random numbers
`rand(n)`	an n by n array of random numbers

List generation/variable indexing

`i:k:l`	list generation: $1stValue : Stride : LastValue$
`v(1)`	1st element of vector v
`v(end)`	last element of vector v
`v(1:2:9)`	1st, 3rd, 5th, 7th, 9th elements of vector v
`v(2:3:9)`	2nd, 5th, 8th elements of vector v
`A(2,3)`	2'nd row, 3'rd column of matrix A
`A(:,3)`	all elements in column 3
`A(1,:)`	all elements in row 1
`A(1:2:end,:)`	all odd rows of matrix A
`A(1:2,2:4)`	sub-matrix of rows 1 and 2, columns 2 through 4
`A(1,end)`	last element in 1'st row

Vector/Matrix op's (`help arith, help ops`)

`+`	addition		
`-`	subtraction		
`*`	multiplication	`.*`	point-wise multiplication
`/`	left division	`./`	point-wise left division
`\`	right division	`.\`	point-wise right division
`ˆ`	exponentiation	`.ˆ`	point-wise exponentiation
`'`	transpose		

Loops (help lang)

```
for k = vectorOrColumnList
    % MATLAB statements
end

while logicalExpression
    % MATLAB statements
end
```

Note that MATLAB is an interpreted language, and hence loops are slower than internal vector manipulation function. So it is better to avoid loops whenever possible.

if/elseif/else construct (help lang)

```
if logicalExpression1    % Mandatory
    % MATLAB statements
elseif logicalExpression2    % Optional
    % MATLAB statements
elseif logicalExpression3    % Optional
    .
    .
elseif logicalExpressionN    % Optional
    % MATLAB statements
else   % Optional
    % MATLAB statements
end   % Mandatory
```

Logical operators (help relop)

`<`	less than
`<=`	less than or equal
`>`	greater than
`>=`	greater than or equal
`==`	equal
`~=`	not equal
`&`	logical AND
`|`	logical OR
`~`	logical NOT

Script M-files

Sequences of MATLAB commands can be stored in text files with the extension `.m`. The commands can be executed by typing the name of the files (without the extension) or through the file management tools provided by the Command Window menu.

Function M-files

Define a separate file called *functionName*.m with the following form:

```
function [out1,...,outN] = functionName(in1,...,inM)
% functionName: A brief one line description (optional)
% .
% .
% More description (optional)
% .
% .
    % First executable statement
    .
    .
    % Valid executable MATLAB statements and comments
    .
    .
    % Last line
```

The function call is made with the following statement:

```
[out1,out2,...,outN] = functionName(in1,in2,...,inM)
```

Useful in M-files (help general, help lang)

`disp`	display a string
`fprintf`	write data to screen of file
`echo`	toggle command echo
`error`	display message and abort
`input`	prompt for input
`keyboard`	transfer control to keyboard
`pause`	wait for time or user response
`return`	return to caller
`warning`	display warning messages

Figure window control (help graphics)

`clc`	clear the command window
`clf`	clear the figure window
`figure`	start a new figure window
`figure(n)`	make figure with index n active. If n is an integer and figure(n) does not exist, create it
`close`	close current figure window
`close(n)`	close figure with index n
`print -dpdf fileName.pdf`	save the current figure in a pdf file

Plotting (help graph2d, help graph3d)

`contour`	contour plot on a plane
`contour3`	3-D contour plot with displayed depth
`mesh`	3-D mesh surface
`meshc`	combination mesh/contour plot
`meshz`	3-D mesh with curtain
`pcolor`	pseudocolor (checkerboard) plot
`plot`	basic 2D plots
`plot3`	plot lines and points in 3-D space
`surf`	3-D colored surface
`surfc`	combination surf/contour plot
`surfl`	3-D shaded surface with lighting

Plotting annotation (help graph2d, graph3d)

`clabel`	contour plot elevation labels
`colorbar`	display color bar (color scale)
`legend`	graph legend
`title`	graph title
`xlabel`	x-axis label
`ylabel`	y-axis label

More about plotting (help graph2d, graph3d)

`box`	toggle the box display
`colormap`	color look-up table
`grid`	toggle the grid state
`hold`	control multiple plots on a single figure
`shading`	color shading mode, e.g. `flat`, `interp`
`subplot`	control multiple plots in one window
`zoom`	enable mouse-based zooming

Math functions (help elfun, datafun, matfun)

The following functions have their intuitive standard meaning: `abs`, `exp`, `log`, `log10`, `log2`, `sqrt`, `sin`, `asin`, `cos`, `acos`, `tan`, `atan`, `floor`, `ceil`, `round`, `max`, `min`, `mean`, `median`, `norm`, `rank`, `det`, `inv`, `sort`.

Performance monitoring (help timefun)

`tic`, `toc`	stopwatch timer functions
`flops`	counts floating point operations