MATLAB Reference Card I

Variable Assignment

`x=[1,2,3,4...];`	Defines a row vector `x` (horizontal)
`x=[1;2;3;4...];`	Defines a column vector `x` (vertical)
`a:c`	The range of integers `a` … `c`, equivalent to `[a,a+1,...c-1,c]`
`a:b:c`	The range of `a` … `c`, with spacing b, equivalent to `[a,a+b,...c-b,c]`
`linspace(a,c,n)`	The range of `a` … `c` with `n` equally spaced values in between
`zeros(m,n)`	An $m \times n$ matrix of zeros (`m` is vertical columns, `n` is horizontal rows)
`ones(m,n)`	An $m \times n$ matrix of ones
`rand(m,n)`	An $m \times n$ matrix of uniformly distributed random numbers $\in [-1,+1]$
`randn(m,n)`	An $m \times n$ matrix of random numbers from $N(\mu = 0, \sigma = 1)$
`x = 'string'`	Defines `x` as the string `string` ("double quotes" are never used)

Variable Indexing

Vectors

`x(1)`	First element
`x(n)`	nth element
`x(end)`	Last element
`x(1:n)`	First `n` elements
`x(end-n:end)`	Last `n+1` elements
`x([1,3,6])`	Specified list of elements
`x(x>0)`	All elements of `x` greater than 0
`x(x>0 & x<9)`	All elements of `x` between 0 and 9

Matrices

`x(i,j)`	Element at row `i` (vertical indexed) and column `j` (horizontal indexed)
`x(i,:)`	All of Row `i`
`x(:,j)`	All of Column `j`
`x(1:m,:)`	First `m` rows
`x(:,1:n)`	First `n` columns
`x(end,end)`	The last element in the last row
`x(:)`	Transformed full matrix to a column vector (column by column)

Variable Manipulation

`x(n) = [];`	Removes element `n` from variable `x`
`x(:,n) = [];`	Removes the column `n` from matrix `x`
`x'`	The complex conjugate transpose of `x` (matters for imaginary data)
`x.'`	The non-conjugate transpose of `x`
`max(x)` `min(x)`	Greatest element in vector `x` Smallest element in vector `x`
`max(x,[],c)`	The greatest elements in matrix `x` along the cth dimension
`[a,i] = max(x)`	Additionally returns the position `i` of the greatest element in `x`
`sort(x)`	Sorts the elements of `x` in ascending order
`sort(x,c)`	Sorts the elements in matrix `x` along the cth dimension
`unique(x)`	Returns all unique values of `x`, sorted in ascending order
`find(x == a)`	Returns indices where `x` is equal to `a`
`reshape(x,[m,n])`	Returns the data in `x`, reshaped to size `[m,n]` (must have same `numel`)
`cat(c,x,y)`	Concatenates the variables `x` and `y` along the dimension `c`

Variable Information

`length(x)`	Length of vector `x` or longest matrix dimension
`s = size(x)`	If `x` is a 5×4 matrix, `s` becomes the vector `[5,4]`
`size(x,c)`	The size of the cth dimension of `x`
`numel(x)`	The number of elements in `x` (can be any dimension)

Matrix Computations

`a+b`	Adds matrices `a` and `b` together, or any scalar `b` to all elements in `a`
`a-b`	Same, with subtraction
`a.*b`	Element-wise multiplies matrices `a` and `b` (they must be the same size)
`a*b`	Matrix multiplies matrices `a` and `b` (inner dimension must match)
`a./b`	Element-wise divides matrices `a` and `b` (they must be the same size)
`a/b`	Matrix divides, roughly equal to `a*inv(b)`
`a.^b`	Element-wise power operation: `a` to the power of `b`

Math Operations

`sin, cos, tan, asin, acos, atan, log, log10, exp, sqrt, ...`
Standard functions, always element-wise operation

`sum(x)`	Sum of elements
`sum(x,c)`	Sum of elements of `x`, along the dimension `c`
`prod(x)`	Product of elements of `x`
`diff(x)`	Difference between every element of `x` (yields length `n-1`)
`cumsum(x)`	Cumulative sum of the elements in `x`
`mean(x)`	Mean of the elements in `x`
`median(x)`	Median of the elements in `x`
`log(x,b)`	Logarithm of `x` with base `b`
`real(x)`	Real part of all elements in `x`
`imag(x)`	Imaginary part of all elements in `x`
`abs(x)`	Absolute value, or magnitude if `x` is complex
`angle(x)`	Angle in radians of the complex number(s) `x`
`mod(x,b)`	Modulus (remainder) of (x/b)

Constants

`i` or `j`	Imaginary unit `sqrt(-1)`
`pi`	3.1415926535897… Yumm
`Inf`	Infinity (e.g. results from `1/0`)
`NaN`	Not a Number (e.g. results from `0/0`)
`exp(1)`	2.7182818284590… Natural logarithm base

Equalities & Logical Operators

`<`	`<=`	Less than	Less than or equal to
`>`	`>=`	Greater than	Greater than or equal to
`==`	`~=`	Equal to	Not equal to
`&`	`&&`	And (element-wise)	And (single value)
`|`	`||`	Or (element-wise)	Or (single value)
`~`		Not	
`any(...)`		`true` if any result in an element-wise expression is `true`	
`all(...)`		`true` if all results of an element-wise expression are `true`	

MATLAB Reference Card II

Documentation

`help <function>`	Displays a description of the `<function>` and how to use it
`doc <function>`	More detailed information than `help`

Workspace

`cd(str)`	Changes the current directory to the string `str`
`addpath(str)`	Adds the directory `str` to the path (files in `str` are also callable)
`clc`	Clears the command window (not variables)
`who`	Displays a list of variables in the workspace
`clear x`	Deletes the variable `x`
`clear`	Deletes all variables in the workspace
`clearvars -except x`	Deletes all variables in the workspace except the variable `x`
`save(name)`	Saves all variables in the workspace to the file `name.mat`
`save(name,'a'...)`	Saves the variable `a` (and possibly others) to the file `name.mat`
`load(name)`	Loads all variables in the file `name.mat`
`load(name,'a'...)`	Loads the variable `a` (and possibly others) from the file `name.mat`

Programming Constructs

`x = [...];`	Arrays / vectors: All variables by default, any number of dimensions.
`s.x = x;`	Structs: Can group many variables (e.g. `x`) into one (e.g. `s`) using `'.'` notation; structs can also be multidimensional (e.g. `s(2,3).x = 6;`).

```
function [out1,out2,...] = myfun(arg1,arg2,...)
  ...
function ...
```

Functions: can be called in the command line using `<myfun>`. Can have more than one function in a `.m` file, but the first one must have the same name as the file. Functions end implicitly where the next one starts. You must call a function with the same number of input arguments, but can return any number of output arguments (e.g. `out = myfun(x1,x2);` or `[out1,out2] = myfun(x1,x2);`).

```
if (...)
  ...
elseif (...)
  ...
else (...)
  ...
end
```

Conditional statements: same as in `C`. Conditions should be logical (evaluate to `true` / `false`), however, variables can be used, in which case only the value `0` is treated as `false`.

```
switch (...)
  case (...)
    ...
  case (...)
    ...
  otherwise
    ...
end
```

Switch statement: same as in `C`. Cases should be possible values of the expression used at the switch. The use of the default case `otherwise` is optional.

```
for i = 1:n
  ...
end
```

For loop: repeated `n` times, where `i` increases by one each iteration. You can use the variable `i` within the loop, but cannot change its value.

```
while (...)
  ...
end
```

While loop: repeated until the condition specified is reached

Scripting

`<name of script>`	Runs the script `<name of script>.m` (see `functions`)
`<line of code>;`	The `;` suppresses any printed output from `line of code`
`%<line of text>`	The `%` creates a comment: nothing after `%` on this line is executed
`keyboard`	Pauses the execution of the current script and gives the user control
`return` or `dbcont`	Resumes the script after keyboard (before `v2014` / after `v2015`)
`...`	Continues the current line of code on the next line
`CTRL+c`	Emergency stop the current script (must be typed in the command line)

Formatting Output

`fprintf(fmt,vars...)`	Like the `C` function `printf`, prints to screen
`sprintf(fmt,vars...)`	Like the `C` function `printf` but prints to a string
`error(msg)`	Displays the string message `msg` and halts execution of the script
`warning(msg)`	Like `error`, but the program continues

Figures

`h = figure(n)`	Creates a new figure number `n` and sets `h` as the handle to it
`h = gcf`	Get current figure handle: same as `h=figure(...)`
`h = subplot(m,n,k)`	Divides a figure into $m \times n$ axes and assigns `h` to the `k`th subplot
`h = gca`	Get current axes: same as `h=subplot(...)`
`get(h)`	With `h` from above, displays all the figure/axes properties
`set(h,'<prop>',x)`	With `h` from above, sets the value of property `<prop>` to `x`
`hold('on')`	All subsequent plots will be added to the current axes
`hold('off')`	Subsequent plots will be overwrite the current axes (default)
`print -depsc2 f1.eps`	Saves the current figure to the file `f1.eps`
`close(n)`	Closes the figure number `n`
`close('all')`	Closes all open figures

Plotting

`plot(y)`	Plot the values of `y` versus `1:length(y)`
`plot(x,y)`	Plot the values of `y` versus `x`
`h = plot(...)`	Returns a handle to the axes used by `plot`
`stem(y)`	Similar to `plot(y)`, but points are shown as disconnected "stems"
`hist(x)`	Bar graph of the histogram of `x`
`hist(x,n)`	Bar graph of the histogram of `x`, using `n` equally distributed intervals
`title(str)`	Sets the title of the current axes to the string `str`
`xlabel(str)`	Label the x-axis with `str` (same for `ylabel`)
`xlim([xmin,xmax])`	Set limits of the x-axis to `xmin` and `xmax` (same for `ylim`)

Images

`I = imread(str)`	Read the image from the filename `str`
`imshow(I)`	Show the image `I` using the default settings
`imshow(I,[])`	Show the image `I` so that `max(I(:))` is white and `min(I(:))` is black
`imshow(I,hot)`	Show the image `I` using the colourmap `hot` (others: `gray`, `jet`, `hsv`,...)
`imwrite(I,str,fmt)`	Write the image `I` to the file named `str`, with format `fmt`