Matlab Cheat Sheet

Some nifty commands

`clc`	Clear command window
`clear`	Clear system memory
`clear x`	Clear x from memory
`ans`	Last result
`close all`	closes all figures
`close(H)`	closes figure H
`whos`	lists data structures
`winopen(pwd)`	Open current folder
`class(obj)`	returns objects class
`int16(x)=y`	convert doubles to Integers
`dlmread('path')`	Reads data
`dlmwrite('path',M)`	Writes M to path
`save filename`	saves all variables to .mat file
`save filename x,y`	saves x,y variables to .mat file
`save -append filename x`	appends x to .mat file
`load filename`	loads all variables from .mat file
`ver`	Lists version and toolboxes
`beep`	Makes the beep sound
`doc function`	Help/documentation for function
`docsearch string`	search documentation
`web google.com`	opens webadress
`inputdlg`	Input dialog box

Portions of matrices and vectors

`x(:)`	All elements of x
`x(j:end)`	j'th to end element of x
`x(2:5)`	2nd to 5th element of x
`x(j,:)`	all j row elements
`x(:,j)`	all j column elements
`diag(x)`	diagonal elements of x
`[A,B]`	concatenates horizontally
`[A;B]`	concatenates vertically

Keyboard shortcuts

`edit filename`	Opens filename in editor
`Alt`	Displays hotkeys
`F1`	Help/documentation for highlighted function
`F5`	Run code
`F9`	Run highlighted code
`F10`	Run code line
`F11`	Run code line, enter functions
`Shift+F5`	Leave debugger
`F12`	Insert break point
`Ctrl+Page up/down`	Moves between tabs
`Ctrl+shift`	Moves between components
`Ctrl+C`	Interrupts code
`Ctrl+D`	Open highlighted codes file
`Ctrl+ R/T`	Comment/uncomment line
`Ctrl+N`	New script
`Ctrl+W`	Close script
`Ctrl+shift+d`	Docks window
`Ctrl+shift+u`	Undocks window
`Ctrl+shift+m`	max window/restore size

Built in functions/constants

`abs(x)`	absolute value
`pi`	3.1415...
`inf`	∞
`eps`	floating point accuracy
`1e6`	10^6
`sum(x)`	sums elements in x
`cumsum(x)`	Cummulative sum
`prod`	Product of array elements
`cumprod(x)`	cummulative product
`diff`	Difference of elements
`round/ceil/fix/floor`	Standard functions..

`*Standard functions: sqrt, log, exp, max, min, Bessel`

`*Factorial(x) is only precise for` $x < 21$

Cell commands

A cell can contain any variable type.

`x=cell(a,b)`	a ×b cell array	
`x{n,m}`	access cell n,m	cellfun
`cell2mat(x)`	transforms cell to matrix	
`cellfun('fname',C)`	Applies fname to cells in C	

Strings and regular expressions

`strcomp`	compare strings (case sensitive)
`strcompi`	compare strings (not case sensitive)
`strncomp`	as strcomp, but only n first letters
`strfind`	find string within a string , gives start position
`regexp`	Search for regular expression

Logical operators

`&&`	Short-Circuit AND.
`&`	AND
`||`	Short-Circuit or
`|`	or
`~`	not
`==`	Equality comparison
`~=`	not equal
`isa(obj, 'class_name')`	is object in class

`*Other logical operators: <,>,>=,<=`

`*All` above `operators are` elementwise

`*Class indicators: isnan, isequal, ischar, isinf, isvector , isempty, isscalar, iscolumn`

`*Short circuits (SC) only evaluate second criteria if first criteria is passed, it is therefore faster.`
`And useful fpr avoiding errors occuring in second criteria`

`*non-SC are bugged and short circuit anyway`

Variable generation

`j:k`	row vector `[j,j+1,...,k]`
`j:i:k`	row vector `[j,j+i,...,k],`
`linspace(a,b,n)`	n points linearly spaced and including a and b
`NaN(a,b)`	a×b matrix of NaN values
`ones(a,b)`	a×b matrix of 1 values
`zeros(a,b)`	a×b matrix of 0 values
`meshgrid(x,y)`	2d grid of x and y vectors
`[a,b]=deal(NaN(5,5))`	declares a and b
`global x`	gives x global scope

Standard Matrix and vector operations

`x=[1, 2, 3]`	1x3 (Row) vector defined
`x=[1; 2; 3]`	3x1 (Column) vector defined
`x=[1, 2; 3, 4]`	2x2 matrix
`x(2)=4`	change index value nr 2
`x.*y`	Element by element multiplication
`x./y`	Element by element division
`x+y`	Element by element addition
`x-y`	Element by element subtraction
`A^n`	normal/Matrix power of A
`A.^n`	Elementwise power of A
`A'`	Transpose
`inv(A)`	Inverse of matrix
`size(x)`	Rows and Columns
`eye(n)`	Identity matrix
`sort(A)`	sorts vector from smallest to largest
`eig(A)`	Eigenvalues and eigenvectors

`*Standard operations: rank,rref,kron,chol`

*Inverse of matrix inv(A) should almost never be used, use RREF through \ instead: $inv(A)b = A\backslash b$.

Matrix and vector operations/functions

`x(x>5)=0`	change elemnts >5 to 0
`x(x>5)`	list elements >5
`find(A>5)`	Indices of elements >5
`find(isnan(A))`	Indices of NaN elements
`B=repmat(A,m,n)`	Makes B from A
`bsxfun(fun,A,B)`	Binary operation on two arrays
`arrayfun(fun,A1,...,An)`	Calls function m times, gets n inputs m times from arrays

`*if arrayfun/bsxfun is passed a gpuArray, it runs on GPU.`

Statistical commands

`hist(x)`	histogram
`distrnd`	random numbers from dist
`distpdf`	pdf from dist
`distcdf`	cdf dist
`distrnd`	random numbers from dist
`distpdf`	pdf from dist
`distcdf`	cdf dist

`*Standard distributions (dist): norm, t, f, gam, chi2, bino`

`*Standard functions: mean,median,var,cov(x,y),corr(x,y),`

`*quantile(x,p) is` not textbook version.

`(It uses interpolation for missing quantiles.`

`*Like most programs, histogram is not a true histogram.`

Structures

`StructName.FieldName =`	Makes structure, and variable named fieldname. Sets value to struct, cell vector or a structure.
`StructName(2).FieldName`	Second element of structure
`getfield(StructName,'FieldName')`	Gets data from structure with fieldname

Plotting commands

`plot(x,y,’Linewidth’,2)`	plots x,y points
`grid`	adds gridlines
`set(gca, ’FontSize’, 14)`	all fonts to size 14
`mesh(x,y,z)`	plots x,y,z points
`figure`	new figure window
`figure(j)`	graphics object j
`get(j)`	returns information graphics object j
`subplot(a,b,c)`	Used for multiple figures in single plot
`xlabel(’\mu line’,’FontSize’,14)`	names x/y/z axis
ylim([a b])	Sets y/x axis limits for plot to a-b
`title(’name’,’fontsize’,22)`	names plot
`grid on;`	Adds grid to plot
`legend(’x’,’y’,’Location’,’Best’)`	adds legends
`hold on`	retains current figure when adding new stuff
`hold off`	restores to default (no hold on)
`set(h,’WindowStyle’,’Docked’);`	Docked window style for plots
`fill`	usefull for coloring polygons
`datetick(’x’,yy)`	time series axis
`semilogx(x,y)`	plot x on log scale
`semilogy(x,y)`	plot y on log scale
`loglog(x,y)`	plot y,x on log scale

For printing figure h to .eps files use:

```
print(figure(h),’-depsc2’,’path\image.eps’)
```

Output commands

`format short`	Displays 4 digits after 0
`format long`	Displays 15 digits after 0
`disp(x)`	Displays the string x
`disp(x)`	Displays the string x
`num2str(x)`	Converts the number in x to string
`num2str([’nA is = ’ num2str(a)])`	OFTEN USED! !
`mat2str(x)`	Converts the matrix in x to string
`int2str(x)`	Converts the integer in x to string
`sprintf(x)`	formated data to a string

System commands

`addpath(string)`	adds path to workspace
`genpath(string)`	gets strings for subfolders
`pwd`	Current directory
`mkdir`	Makes new directory
`tempdir`	Temporary directory
`inmem`	Functions in memory
`exit`	Close matlab
`dir`	list folder content
`ver`	lists toolboxes

Nonlinear nummerical methods

`quad(fun,a,b)`	simpson integration of @fun from a to b
`fminsearch(fun,x0)`	minimum of unconstrained multivariable function using derivative-free method
`fmincon`	minimum of constrained function

Example: Constrained log-likelihood maximization, note the -

```
 Parms_est = fmincon(@(Parms) -flogL(Parms,x1,x2,x3,y)
 ,InitialGuess,[],[],[],[],LwrBound,UprBound,[]);
```

Debbuging etc.

`keyboard`	Pauses exceution
`return`	resumes exceution
`tic`	starts timer
`toc`	stops timer
`profile on`	starts profiler
`profile viewer`	Lets you see profiler output
`try/catch`	Great for finding where errors occur
`dbstop if error`	stops at first error inside try/catch block
`dbclear`	clears breakpoints
`dbcont`	resume execution
`lasterr`	Last error message
`lastwarn`	Last warning message
`break`	Terminates executiion of for/while loop
`waitbar`	Waiting bar

Data import/export

`xlsread/xlswrite`	Spreadsheets (.xls,.xlsm)
`readtable/writetable`	Spreadsheets (.xls,.xlsm)
`dlmread/dlmwrite`	text files (txt,csv)
`load/save -ascii`	text files (txt,csv)
`load/save`	matlab files (.m)
`imread/imwrite`	Image files

Programming commands

`return`	Return to invoking function
`exist(x)`	checks if x exists
`G=gpuArray(x)`	Convert varibles to GPU array

```
function [y1,...,yN] = myfun(x1,...,xM)
```

Anonymous functions not stored in main programme

```
myfun = @(x1,x2) x1+x2;
```

or even using

```
myfun2 = @myfun(x) myfun(x3,2)
```

Conditionals and loops

```
  for i=1:n
     procedure   Iterates over procedure
  end            incrementing i from 1 to n by 1


while(criteria)
   procedure     Iterates over procedure
end              as long as criteria is true(1)
```

```
if(criteria 1)              if criteria 1 is true do procedure 1
   procedure1
elseif(criteria 2)          ,else if criteria 2 is true do procedure 2
   procedure2
else                        , else do procedure 3
   procedure3
end

switch switch_expression    if case n holds,
case 1                      run procedure n. If none holds
    procedure 1             run procedure 3
case 2                      (if specified)
    procedure 2
otherwise
    procedure 3
end
```

General comments

- Monte-Carlo: If sample sizes are increasing generate longest size first in a vector and use increasingly larger portions for calculations.
- Trick: Program that (1) takes a long time to run and (2) doesnt use all of the CPU/memory ? - split it into more programs and run using different workers (instances).
- Matlab is a column vector based language, load memory columnwise first always.
- Matlab uses copy-on-write, so passing pointers (adresses) to a function will not speed it up.
- You can turn the standard (mostly) Just-In-Time compilation off using: feature accel off. You can use compiled (c,c++,fortran) functions using MEX functions.
- For faster code also prealocate memory for variables, Matlab requires contiguous memory usage!.
- Some excellent toolboxes: MFE toolbox (Econometrics).
- Functions defined in a .m file is only available there, give own file if they are used otherplaces and name them as myfun.m if called myfun in definition.
- Graphic cards(GPU)’s have many (small) cores. If (1) program is computationally intensive (not spending much time transfering data) and (2) massively parallel, so computations can be independent. Consider using the GPU!
- Using multiple cores (parallel computing) is often easy to implement, just use parfor instead of for loops.
- Warnings: empty matrices are NOT overwritten $([] + 1 = [])$. Rows/columns are added without warning if you write in a nonexistent row/column. Good practise: Use 3i rather than 3*i for imaginary number calculations, because i might have been overwritten by earlier. 1/0 returns inf, not NaN. Dont use == for comparing doubles, they are floating point precision for example: $0.01 == (1 - 0.99) = 0$.

`http://www.econ.ku.dk/pajhede/`