Matlab Cheat Sheet

Some nifty commands

`clc`	Clear command window
`clear`	Clear system memory
`clear x`	Clear x from memory
`commandwindow`	open/select commandwindow
`whos`	lists data structures
`whos x`	size, bytes, class and attributes of x
`ans`	Last result
`close all`	closes all figures
`close(H)`	closes figure H
`winopen(pwd)`	Open current folder
`class(obj)`	returns objects class
`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
`methods(A)`	list class methods for A

Statistical commands

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

`*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.`

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
`cell2mat(x)`	transforms cell to matrix
`cellfun('fname',C)`	Applies fname to cells in C

cellfun

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 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

Tables

`T=table(var1,var2,...,varN)`	Makes table*
`T(rows,vars)`	get sub-table
`T{rows,vars}`	get data from table
`T.var or T.(varindex)`	all rows of var
`T.var(rows)`	get values of var from rows
`summary(T)`	summary of table
`T.var3(T.var3>5)=5`	changes some values
`T.Properties.Varnames`	Variable names
`T = array2table(A)`	! make table from array
`T = innerjoin(T1,T2)`	innerjoin
`T = outerjoin(T1,T2)`	outerjoin !

`Rows and vars indicate rows and variables.`
`tables are great for large datasets, because they use less memory and allow faster operations.`
`*rowfun is great for tables, much faster than eg. looping`

matrix and vector operations/functions

`x=[1, 2, 3]`	1x3 (Row) vector
`x=[1; 2; 3]`	3x1 (Column) vector
`x=[1, 2; 3, 4]`	2x2 matrix
`x(2)=4`	change index value nr 2
`x(:)`	All elements of x (same as x)
`x(j:end)`	j'th to last 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
`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
`numel(A)`	number of array elements
`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
`[A,B]`	concatenates horizontally
`[A;B]`	concatenates vertically

`For functions on matrices, see bsxfun,arrayfun or repmat`
`*if arrayfun/bsxfun is passed a gpuArray, it runs on GPU.`
`*Standard operations: rank,rref,kron,chol`
*Inverse of matrix inv(A) should almost never be used, use RREF through \ instead: $inv(A)b = A\backslash b$.

Plotting commands

`fig1 = plot(x,y)`	2d line plot, handle set to fig
`set(fig1, 'LineWidth', 2)`	change line width
`set(fig1, 'LineStyle', '-')`	dot markers (see *)
`set(fig1, 'Marker', '.')`	marker type (see *)
`set(fig1, 'color', 'red')`	line color (see *)
`set(fig1, 'MarkerSize', 10)`	marker size (see *)
`set(fig1, 'FontSize', 14)`	fonts to size 14
`figure`	new figure window
`figure(j)`	graphics object j
`get(j)`	returns information graphics object j
`gcf(j)`	get current figure handle
`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/off;`	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
`datetick('x',yy)`	time series axis
`plotyy(x1,y1,x2,y2)`	plot on two y axis
`refreshdata`	refresh data in graph if specified source
`drawnow`	do all in event queue

```
* Some markers: ', +, *, x, o, square
* Some colors:   red, blue, green, yellow, black
* color shortcuts:   r, b, g, y, k
* Some line styles:   -, --, :, -.
* shortcut combination example:   plot(x,y,'b--o')
```

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

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

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

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

General comments

- Monte-Carlo: If sample sizes are increasing generate largest size first in a vector and use increasingly larger portions for calculations. Saves time+memory.
- 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. For faster code also prealocate memory for variables, Matlab requires contiguous memory usage!. Matlab uses copy-on-write, so passing pointers (adresses) to a function will not speed it up. Change variable class to potentially save memory (Ram) using: int8, int16, int32, int64, double, char, logical, single
- 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.
- Avoid global variables, they user-error prone and compilers cant optimize them well.
- Functions defined in a .m file is only available there. Preface function names with initials to avoid clashes, eg. MrP_function1.
- 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: $0.01 == (1-0.99) = 0$.