Matlab cheat-sheet — Pedro Fortuny Ayuso — Uniovi — April 24, 2015

Basics

`[a:s:b]` — Vector from `a` to `b` in steps of length `s`.
`linspace(a,b,n)` — Vector of `n` even-spaced numbers between `a` and `b`.
`pi, exp(1)` — Values of π and e (3.14159... and 2.71828...).
`clear, clear(x)` — Clear the values of all variables or only of `x`.
`whos, who` — Describe or list the defined variables.

Arithmetic and Logic operators

`+ - * /` — Basic operations with numbers and matrices.
`^` — Raise to a power a matrix: `a^b`= a^b.
`.* ./ .^` — Multiply, divide and rise to a power *elementwise*.
`& | ~` — Logical operators **and** (`&`), **or** (`|`), **not** (`~`).
`all, any` — Test whether *all* (`all`) or *any of* (`any`) the elements of a vector satisfy a condition: `any(x>0)` Is any of the elements of `x` greater than 0?

Matrices and vectors

`[a:s:b]` — Vector from `a` to `b` in steps of length `s`.
`linspace(a,b,n)` — Vector of `n` even-spaced numbers between `a` and `b`.
`length(v)` — Length of vector `v`.
`size(M)` — Size (`rows`×`columns`) of `M`.
`zeros(n,m)` — Matrix full of 0s with `n` rows and `m` columns. If `m` is missing, square matrix `n`×`n`.
`ones(n,m)` — Matrix full of 1s with `n` rows and `m` columns.
`eye(n)` — Identity matrix of size `n`.
`eye(n,m)` — An `n`×`m` matrix such that the leftmost `n`×`n` submatrix is the identity and the rest is full of 0.
`diag(v)` — Square matrix whose diagonal is the vector `v`.
`diag(v,n)` — Square matrix whose `n`-diagonal is the vector `v`. The 0-diagonal is the principal. If `n`<0, count downwards, if `n`>0, count up.
`rand(n,m)` — Random matrix of size `n`×`m`. The values are evenly distributed between 0 and 1.

Accessing elements

If `x` is a vecto

`x(i)` — Element `i`-th of `x`.
`x(i:j)` — Elements from the `i`-th to the `j`-th of vector `x`.
`x(i:end)` — Elements from the `i`-th to the last one of vector `x`.
`x(:)` — All the elements of `x`.

If `A` is a matrix

`A(i,j)` — Element `i,j` of `A`.
`A(a:b,c:d)` — Elements in rows from `a` to `b` which are in columns from `c` to `d` (i.e. a *submatrix*).
`diag(A)` — Elements in the principal diagonal of `A`.

Functions and utilities

`sqrt` — Square root. `sqrt(x)`: square root of each element of x.
`sin, cos, tan` — Sine, cosine and tanget. For example, `sin(x)`, sine of each element of `x`.
`asin, acos, atan` — Inverse functions of the previous ones.
`exp` — Exponential. `exp(x)`: exponential of *each element* of `x`.
`log, log10` — Natural logarithm (base e) and decimal (base 10). `log(x)`: natural logarithm of each element of `x`.

Function definition

`f = @(x) sin(x) - exp(x) .* x.^2` — Defines the one-variable function `f` whose value is $\sin(x) - e^x x^2$ (always use `.*`, `./` y `.^`).
`f = @(x,y,z) x .* y - z.^2 .* y ./ x` — Defines the three-variable function `t` whose value is $xy - z^2 y/x$. (always use `.*`, `./` y `.^`).
`function [z] = patata(a, b, c) ..... end` — Inside file `patata.m`, definition of the function `patata` which returns *one* value `z` and requires 3 parameters (`a`, `b` and `c`).
`function [Sol N M] = pototo(a, b, c) ..... end` — Inside file `pototo.m`, definition of the function `pototo` which returns *three* values (`Sol`, `N` and `M`) and requires three parameters (`a`, `b` and `c`).

Plotting

`clf` — Clears the graphical window.
`hold on/hold off` — Allows (`hold on`) or prevents (`hold off`) plotting on the same graph.
`plot(y)` — If `y` is a vector, plot the sequence of values of `y`.
`plot(x,y)` — If `x` and `y` are vectors *of the same length*, plot the graph of `x` against `y` given by those values.
`plot(x,f(x))` — If `x` is a vector and `f` a function, plot the points `(x,f(x))`.
`fplot(f,[x0 x1 y0 y1])` — If `f` is a function, plot `f` on the rectangle `[x0, x1]`×`[y0, y1]`.
`axis([x0 x1 y0 y1])` — Replot a graph on the rectangle `[x0, x1]`×`[y0, y1]`.
`xlim([x0 x1]), ylim([y0 y1]` — Replot a graph with `x` varying between `x0` and `x1` (or with `y` between `y0` and `y1`).

Matlab cheat-sheet — Pedro Fortuny Ayuso — Uniovi — April 24, 2015

Programming

`return` — Return *immediately* from a function.

```
if(COND)
   ...
end
```
— If `COND` holds, perform the instructions `...`.

```
if(COND)
   ...1
else
   ...2
end
```
— If `COND` holds, do `...1`; if it does not, do `...2`.

```
for k = a:s:b
   ...
end
```
— Loop from `a` to `b` in steps of `s`, in the variable `k`.

```
for k = v
   ...
end
```
— If `v` is a vector, loop in the variable `k` traversing the elements of `v`.

```
while(COND)
   ...
end
```
— Repeat the instructions in `...` while `COND` holds.

Assuming `nombreF.m` is a `.m` file

```
function [x1 x2] = nombreF(a,b,c)
   ...
end
```
— Defines function `nombreF`, which as, as input parameters, `a`, `b` y `c` and returns values `x1` y `x2`.

Advanced matrix operations

`hilb(n)` — Hilbert matrix of order `n`. Each entry (i, j) values $1/(i + j)$.
`rref(M)` — *Reduced echelon* matrix associated to `M`.
`cond(M)` — Condition number of `M`.
`inv(M)` — Inverse matrix *of a square matrix* (if it exists).
`\` — Solution of a linear system: `A \ b` solves the system $Ax = b$.
`[A B] = lu(M)` — LU factorization of `M`: On one hand, `AB` = `M` and on the other, `A` is lower triangular and `B` is upper triangular.
`chol(A)` — Cholesky factorization of `A` (for `A` symmetric and positive definite). A matrix `R` such that `R'*R`=`A`.

Polynomial operations

Assuming `v` is a vector `v=[`$\mathtt{v}_1, \ldots, \mathtt{v}_n$`]`.

`polyval(v,x)` — Value of the polynomial expression $\mathtt{v}_1 x^{n-1} + \cdots + \mathtt{v}_{n-1} x + \mathtt{v}_n$ (the degree *decreases*).
`roots(v)` — Roots of the polynomial $\mathtt{v}_1 x^{n-1} + \cdots + \mathtt{v}_{n-1} x + \mathtt{v}_n$.
`polyderiv(v)` — Derivative (as a vector expression) of the polynomial $\mathtt{v}_1 x^{n-1} + \cdots + \mathtt{v}_{n-1} x + \mathtt{v}_n$, that is, $[\mathtt{v}_1/(\mathtt{n}-1), \ldots, \mathtt{v}_{n-2}/2, \mathtt{v}_{n-1}]$.

Numerical operations

Assuming `v` is a vector `v=[`$\mathtt{v}_1, \ldots, \mathtt{v}_n$`]`.

`sum(v), prod(v)` — Sum and product of the elements of `v`.
`max(v), min(v)` — Maximum and minimum of the elements of `v`.
`mean(v), var(v)` — Mean and variance of the elements of `v`.
`diff(v)` — Successive differences of `v`. That is, $[\mathtt{v}_2 - \mathtt{v}_1, \mathtt{v}_3 - \mathtt{v}_2, \ldots, \mathtt{v}_n - \mathtt{v}_{n-1}]$. One has that `length(diff(v))`=`length(v)-1`.
`diff(v,k)` — Successive `k`-*differences* of `v`. One has `length(diff(v,k))`=`length(v)`

Interpolation

`interp1(x,y)` — Given `x` and `y` two equally-sized vectors of length n, returns the interpolation polynomial (Lagrange) for the points (x, y). It is returned as a vector $[\mathtt{v}_1, \ldots, \mathtt{v}_n]$ (interpreted as a polynomial of degree $n - 1$).
`polyfit(x,y,n)` — Polynomial (as a vector) of degree `n` which minimizes the quadratic error with respect to the cloud of points given by `x` and `y` (equally-sized vectors).
`spline(x,y)` — Cubic spline interpolation function for the points (x, y) given by `x` and `y` (equally-sized vectors). It returns a *piecewise polynomial* which can be evaluated using `ppval`.