Matlab cheat sheet for Calculus - Pedro Fortuny Ayuso - Uniovi - September 23, 2013

Basics

`[a:s:b]` — Vector from `a` to `b` in steps of length `s`.
`linspace(a,b,n)` — Vector of `n` equidistributed elements from `a` to `b`.
`pi, exp(1)` — Values of π and e ($3.14159\dots$, $2.71828\dots$).
`clear, clear(x)` — Clear all variables, or only `x`.
`whos, who` — Describe or list all the defined variables.
`double(x)` — Express `x` in floating point.

Arithmetic and logical operations

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

Vectors (and matrices)

`linspace(a,b,n)` — Vector of `n` equidistributed numbers between `a` and `b`.
`[a:s:b]` — Vector of numbers from `a` to `b` in steps of length `s`.
`length(v)` — Length (number of elements) of vector `v`.
`size(M)` — Size (`rows`×`columns`) of `M`.
`zeros(n,m)` — Matrix full of 0s of `n` rows and `m` columns. If `m` is missing, square matrix of size `n`×`n`.
`ones(n,m)` — Matrix full of 1s of `n` rows and `m` columns.
`eye(n)` — Identity matrix of rank `n`.
`eye(n,m)` — Matrix of size `n`×`m` such that the main diagonal is full of 1s and the other elements are 0.
`diag(v)` — Square diagonal matrix with `v` in the diagonal.
`rand(n,m)` — Random matrix of `n` rows and `m` columns. The elements are between 0 and 1.

Accessing elements of vectors (matrices)

If `x` is a vector

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

If `A` is a matrix

`A(m,n)` — Element `m,n` of `A`.
`A(a:b,c:d)` — Submatrix from row `a` to row `b` and from col. `c` to col. `d` of `A`.
`A(a,:), A(:,b)` — The whole row `a` or the whole column `b` of `A`.
`diag(A)` — Elements on the main diagonal of `A`.

Functions and utilities

`sqrt` — Square root. `sqrt(x)`: square root of every element of `x`.
`sin, cos, tan` — Sine, cosine and tangent: `sin(x)`, sine of every element of `x`.
`asin, acos, atan` — Inverse trigonometrical functions.
`exp, log, log10` — Exponential, logarithm and base 10 logarithm. So, `exp(x)`: exponential of every element of `x`.

Function Definition

`f = @(x) sin(x) - exp(x) .* x.^2` — Defines function `f`, of one variable, whose value for x is $\sin(x) - e^x x^2$ (Always use `.*`, `./` and `.^`).
`f = @(x,y,z) x .* y - z.^2 .* y ./ x` — Defines function `f` of 3 variables, whose value for (x,y) is $xy - z^2 y/x$. (Always use `.*`, `./` and `.^`).
`function [z] = potato(a, b, c) ...... end` — *Inside* file `potato.m`, defines `potato`, which returns one value `z` and requires 3 parameters (`a`, `b` and `c`).
`function [Sol N M] = tomato(a, b) ...... end` — *Inside* file `tomato.m`, defines `tomato`, returning `[Sol, N, M]` and requiring two parameters (`a` and `b`).

Plots

`clf, cla` — Clear the graphical window (`clf`) or the active figure (`cla`).
`hold on/hold off` — Turns on/off the overplotting toggle: `hold on` turns it on, `hold off` turns it off.
`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 values of `y` against those of `x`.
`plot(x,f(x))` — If `x` is a vector and `f` a function, plot the points `(x,f(x))`.
`subplot(n,m,i)` — Divide the graphic screen into `n` rows `m` and columns and select the `i`-th one for plotting on it at the next call of `plot`.
`ezplot(f(x))` — If `f(x)` is a function of a variable, plot its graph.
`ezplot(f(x,y))` — If `f(x,y)` is a function of two variables, plot the set $f(x,y) = 0$.
`axis([x0 x1 y0 y1])` — Redraw all the plots using the rectangle `[x0, x1]`×`[y0, y1]`.

`xlim([x0 x1]), ylim([y0 y1])` — Redraw a plot for `x` between `x0` and `x1` (or for `y` between `y0` and `y1`).

Symbolic Computations

`syms x y t` — Declare the variables `x`, `y` and `t` as symbolic.

`f=x.^2+cos(y)` — Define `f` as the symbolic function $x^2+\cos(y)$ in the variables x, y contained in the expression. Previously, `syms x y` should have been run.

`g=matlabFunction(f)` — If `f` is a symbolic function, define `g` as the equivalent anonymous ("with `@`") one.

`solve(expr, x)` — If `expr` is a symbolic expression in the variable `x`, solves the equation $expr = 0$. Returns a *column vector* containing a solution in each row. These can be given in floating point or as exact expression (use `double` for the values). Ex.: `syms x; solve(x.^2 - 1, x);`

`limit(f,x,a,'right') - limit(f,x,a,'left')` — Right and left limits of `f` as x tends to a. The variable x must have been declared as `syms`, while `a` must be a number.

`diff(f,x,n)` — n−th derivative of `f` with respect to `x`. The variable `x` must have been declared `syms` and `n` must be a natural number.

`int(f,x)` — Primitive of `f` with respect to the symbolic variable `x`.

`int(f,x,a,b)` — Definite integral of `f` between `a` and `b` (i.e. $\int_a^b f(x)\,dx$). The variable `x` is the one in `f` and must have been declared symbolic.

`taylor(f,x,a,n) - taylor(f,a,n) - taylor(f,n)` — Taylor expansion of `f` with respect to the variable `x`, at `a`, of order `n`.

Advanced matrix operations

`inv(M)` — Inverse matrix of *square matrix* `M` (if it exists).

`\` — Solve a linear system: `A \ b` solves the system $Ax = b$ (approximately).

`[A B] = lu(M)` — LU factorization of matrix `M`: that is, the returned values satisfy $\mathtt{AB} = \mathtt{M}$, `A` is lower triangular with 1 on its diagonal and `B` is upper triangular.

Polynomial operations

In the following commands, `v` is a vector: `v=[`$\mathtt{v}_1, \dots, \mathtt{v}_n$`]`.

`polyval(v,a)` — Value of the polynomial expression $\mathtt{v}_1 a^{n-1} + \cdots + \mathtt{v}_{n-1} a + \mathtt{v}_n$ (notice that the exponents go from $n-1$ to 0).

`roots(v)` — (Approximate) 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 representing a polynomial) of $\mathtt{v}_1 x^{n-1} + \cdots + \mathtt{v}_{n-1} x + \mathtt{v}_n$, that is, $[(n-1)\mathtt{v}_1, \dots, 2\mathtt{v}_{n-2}, \mathtt{v}_{n-1}]$.

`polyint(v)` — Integral (as a vector representing a polynomial) of $\mathtt{v}_1 x^{n-1} + \cdots + \mathtt{v}_{n-1} x + \mathtt{v}_n$, that is, $[\mathtt{v}_1/(\mathtt{n}-1), \dots, \mathtt{v}_{n-2}/2, \mathtt{v}_{n-1}, 0]$: the constant is 0.

Numerical computations

In the following commands, `v` is a vector: `v=[`$\mathtt{v}_1, \dots, \mathtt{v}_n$`]`.

`sum(v), prod(v)` — Sum and product of the elements of `v`. `v`.

`max(v), min(v)` — Maximum and minimum of the elements of `v`.

`mean(v), var(v)` — Mean and variance of the elements `v`.

`diff(v)` — Successive differences of `v`. That is, the vector $[\mathtt{v}_2 - \mathtt{v}_1, \mathtt{v}_3 - \mathtt{v}_2, \dots, \mathtt{v}_n - \mathtt{v}_{n-1}]$. Notice that `length(diff(v))=length(v)-1`.

`diff(v,k)` — `k`-*successive differences* of `v` (iteration of the previous command). Notice that `length(diff(v,k))=length(v)-k`.

Interpolation &c

In the following commands, `u` and `v` are vectors of the same length.

`interp1(u, v)` — Lagrange interpolating polynomial for the cloud of points (u, v), as a vector $[\mathtt{w}_1, \dots, \mathtt{w}_n]$ (which represents as a polynomial of degree $n-1$).

`polyfit(x,y,n)` — Polynomial of degree `n` minimizing the quadratic error with respect to the cloud of points (u, v).

`spline(x,y)` — Cubic spline for the cloud of points (u, v). The value returned is a *piecewise-defined polynomial* which should be evaluated using `ppval`.

`ppval(p, x)` — Evaluate the *piecewise* polynomial `p` on vector `x`. A piecewise polynomial is a special structure which, for example, is returned by `spline()`.