# Aide-mémoire résumé de Maple

## Symboles - Abréviations - Environnement

| | Description | Exemple |
|---|---|---|
| `restart` | efface toutes les définitions, libère la mémoire | `restart:` |
| `;  :` | fin d'instruction avec/sans affichage | `int( x^2, x );  int( x^2, x ):` |
| `with` | charge un package Maple | `with( linalg ); with( plots ):` |
| `?` | accès à l'aide en ligne de Maple | `?plots` |
| `:=` | affectation d'une variable | `expr := x^2/y^3;` |
| `assign` | affectation<br>(souvent utilisé après `solve` ou `dsolve`) | `S:=solve( {x+y=1, 2*x+y=3}, {x,y} );`<br>`assign( S ); x; y;` |
| `` ` ` `` | délimiteur de nom de variable | `` `Ma variable` := 3*cos(Pi/8); `` |
| ´ ´ | empêche l'évaluation | `x := ´x´;` |
| `"  "` | délimiteur de chaine de caractères | `plot( sin(10*x) + 3*sin(x), x=0..2*Pi,`<br>`  title="Un tracé intéressant" );` |
| `..` | définit un intervalle | `plot( t*exp(-2*t), t=0..3 );` |
| `{ }` | délimiteur d'ensemble (éléments non ordonnés) | `{ y, x, y };` |
| `[ ]`<br>`L[i]` | délimiteur de liste (éléments ordonnés)<br>$i^{\text{ème}}$ élément de la liste `L` | `list:=[ y, x, y ];`<br>`list[2]` |
| `seq, $` | créent une séquence | `seq( [0,i], i=-3..3 );  i$i=1..4` |
| `for from to by`<br>`while do od` | répétition d'instructions | `tot := 0;`<br>`for i from 11 by 2 while i < 100 do`<br>`  tot := tot + i^2`<br>`od;` |
| `if then elif then`<br>`else fi` | test conditionnel | `if x>=0 then 1 else -1;` |
| `%` | référence au résultat précédemment évalué<br>(à utiliser avec parcimonie) | `Int( exp(x^2), x=0..1 ):`<br>`% = evalf( % );` |
| `->` | définition d'application (ou de procédure) | `f:=(x,y) -> x^2*sin(x-y); f(Pi/2,0);` |
| `unapply` | définition d'application | `x^2*sin(x-y); f:=unapply(%,x,y);` |
| `assume` | formulation d'une hypothèse sur un objet | `assume( t>0 ); assume( z, real )` |
| `\|\|` | concaténation de chaîne ou nom | `a\|\|i $ i=1..5;` |
| `rhs lhs` | membre de droite/gauche d'une équation | `rhs(x=3*t+1);` |

## Opérations mathématiques, fonctions et constantes

| | Description | Exemple |
|---|---|---|
| `+, -, *, /, ^` | opérations arithmétiques | `3*x^(-4) + x/Pi;` |
| `sin, cos, tan, cot` | fonctions trigonométriques circulaires | `sin( theta-Pi/5 ) - cos( theta^2 );` |
| `arcsin, arccos,`<br>`arctan, arccot` | fonctions trigonométriques circulaires réciproques | `arctan( 2*x );` |
| `sinh, cosh, tanh` | fonctions trigonométriques hyperboliques | `sinh( 0 ) + cosh( 2 ); tanh(ln(2));` |
| `arcsinh, arccosh,`<br>`arctanh` | fonctions trigonométriques hyperboliques réciproques | `arctanh( -1/2 );` |
| `exp, ln, log10` | fct exponentielle, logarithme, log base 10 | `exp( 2*x ); ln( x*y/2 ); log10(1000);` |
| `abs, sqrt` | module (valeur absolue), fonction $\sqrt{\ }$ | `abs( (-3)^5 ); sqrt( 24 );` |
| `floor, ceil` | partie entière, partie entière supérieure | `floor(2.78);` |
| `@  @@` | opérateur $\circ$ de composition (répété) | `(cos@arcsin)(x); (D@@2)(ln);` |
| `!` | factorielle | `100!;` |
| `=, <>, <, <=, >, >=` | équations et inégalités | `exp(Pi) > Pi^exp(1);` |
| `Pi, I, exp(1)` | $\pi$, i, e (constantes mathématiques) | `exp( Pi*I );` |
| `infinity,-infinity` | $+\infty$, $-\infty$ | `limit( x^(-2), x=infinity );` |
| `.` | séparateur décimal, provoque un calcul numérique approché | `3+ln(0.5);` |
| `Digits` | nombre de chiffres significatifs des évaluations numériques | `Digits:=15;` |

## Manipulations d'expressions

| | Description | Exemple |
|---|---|---|
| `limit, diff, int` | calcule la limite, dérivée, intégration ou primitive d'une **expression** | `limit( 1/x , x=0 , left );`<br>`diff( a*x*exp(b*x^2)*cos(c*y), x )`<br>`int( sqrt(x), x=0..Pi );` |
| `Limit, Diff, Int` | forme inerte (non évaluée) des précédentes | `Limit( sin(a*x)/x, x=0 );` |
| `value` | évalue une expression (utilisée habituellement avec `Limit`, `Diff`, ou `Int`) | `G := Int( exp(-x^2), x );`<br>`value( G );` |
| `taylor`<br>`series` | développement limité d'une `expression`<br>développement asymptotique d'une `expr.` | `taylor( (cos(x)-1)/x^2, x=0, 12 );`<br>`series( sin(x)/x^2, x=0, 12 );` |
| `plot` | représentation de graphiques dans le plan<br>options `scaling=constrained`, `colour=blue`, `discont=true`, `thickness=3`... | `plot( u^3, u=0..1, title="cubique" );`<br>`plot( [sin(x), cos(x)], x=0..Pi );`<br>`plot([x(t), y(t),t=a..b],-1..1,-1..1);`<br>`plot([r(t),t,t=a..b],coords=polar);`<br>`plot([r(u),t(u),u=a..b],coords=polar);` |
| `display` | affiche plusieurs graphiques sur une même image (nécessite le package `plots`) | `F:=plot( exp(x), x=0..3, style=line );`<br>`G:=plot( 1/x, x=0..3, style=point );`<br>`display([F,G],title="2 courbes");` |
| `solve` | résolution d'équations, d'inéquations ou de systèmes | `eqs:={x+y=3,x-y=2}; incs:={x,y};`<br>`solve(eqs,incs);` |
| `allvalues` | tente de donner une expression de racines représentées par `RootOf` | `e:=RootOf(_Z^5-1);`<br>`allvalues(e);` |
| `dsolve` | résolution d'équation différentielle ordinaire; voir `?dsolve` pour les options | `Ed := diff( y(x), x$2 ) + y(x) = 1;`<br>`dsolve( Ed, y(x) );` |
| `D`<br>`D[i]` | opérateur de dérivation de **fonction**<br>dérivation par rapport à la ième variable | `CI := y(0)=1, D(y)(0)=1;`<br>`dsolve( { Ed, CI }, y(x) );` |
| `evalf, evalc`<br>`evalb/is, evalm` | évaluation approchée, complexe (`a+I*b`)<br>booléenne (`true` ou `false`), matricielle | `evalf( ln(Pi) ); evalc( exp(x+I*y) );`<br>`evalb( evalf( exp(Pi) > Pi^exp(1) ) );` |
| `subs` | substitution de valeurs dans une expr. | `subs( x=2, 3*x*ln(x^3) );` |
| `simplify, combine` | simplification d'une expression | `simplify( exp( a+ln(b*exp(c)) ) );` |
| `factor` | factorisation d'une expression polynomiale | `factor( (x^3-y^3)/(x^4-y^4) );` |
| `convert` | conversion à une expr. de forme différente | `convert( x^3/(x^2-1), parfrac, x );`<br>`convert( taylor(sin(x),x), polynom );` |
| `collect, sort` | regroupe les coefficients de même puissance | `collect( (x+1)^3*(x+2)^2, x );` |
| `coeffs, coeff` | coefficients d'un polynôme | `coeffs(p,x); coeff(p,x^2);` |
| `numer` | extrait le numérateur d'une expression | `numer( (x+1)^3/(x+2)^2 );` |
| `denom` | extrait le dénominateur d'une expression | `denom( (x+1)^3/(x+2)^2 );` |

## Package LinearAlgebra

| | Description | Exemple |
|---|---|---|
| `Matrix, Vector, << >>`<br>`, \|`<br>`M[i,j]`<br>`V[i]`<br>`M-a`<br>`Equal` | définit une matrice, un vecteur<br>changement de ligne, de colonne<br>coefficient $m_{i,j}$ de matrice<br>coordonnée $i$ de vecteur<br>matrice $\mathtt{M} - \mathtt{a}\mathrm{I}_n$ si $\mathtt{a} \in \mathbb{K}$<br>teste l'égalité entre deux matrices | `A := Matrix([[a,b],[c,d]]);`<br>`B := Matrix(2,2,[1,2,3,4]);`<br>`A[1,2];`<br>`C := < <1, 2> \| <3, 4> >`<br>`Equal(A,IdentityMatrix(3));` |
| `DiagonalMatrix` | définit une matrice diagonale | `DiagonalMatrix([1,1,1]);` |
| `.` ou `Multiply` | multiplication matricielle | `M . X;` |
| `Determinant, MatrixInverse`<br>`Transpose, Trace, Rank` | évident... | `Determinant(A-x);`<br>`Transpose(A);` |
| `NullSpace, ColumnSpace` | base du noyau, de l'image d'une matrice | `NullSpace(A); ColumnSpace(A);` |
| `CrossProduct, DotProduct` | pdt vectoriel, pdt scalaire vecteurs | `CrossProduct(U,V);` |
| `Normalize( ,2), Norm( ,2)` | normalise un vecteur, donne la norme d'un vecteur | `U:=Normalize(U,2); Norm(U,2);` |
| `EigenValues, EigenVectors,`<br>`CharacteristicPolynomial` | valeurs propres, vecteurs propres, polynôme caractéristique | `EigenVectors(A);`<br>`CharacteristicPolynomial(A,x);` |