Mémento wxMaxima

1. *Calculs numérique :*

`sqrt(2)`	retourne la racine carrée de 2. Attention, Maxima utilise les puissances rationnelles pour simplifier les racines carrées : par exemple, `sqrt(8)` sera affichée en $2^{\frac{3}{2}}$
`3^2`	retourne le carré de 3
`6!`	retourne la factorielle de 6

`cos(%pi/4)`
`sin(%pi/4)`
`tan(%pi/4)`
	attention, les arguments des fonctions trigonométriques s'expriment en radians

`abs(-3)`	retourne la valeur absolue de -3
`divide(15,6)`	effectue la division euclidienne de 15 par 6. Cette commande retourne une liste de deux éléments : le quotient et le reste de la division euclidienne.
`mod(5,2)`	retourne le reste de la division euclidienne de 5 par 2
`ceiling(5.3)` $\rightsquigarrow 6$ `ceiling(-5.3)` $\rightsquigarrow -5$	retourne le plus petit des entiers supérieurs à l'argument
`floor(5.3)` $\rightsquigarrow 5$ `floor(-5.3)` $\rightsquigarrow -6$	retourne le plus grand des entiers inférieurs à l'argument
`truncate(-5.3)` $\rightsquigarrow -5$	de l'encadrement à l'unité près de l'argument, cette commande retourne l'entier le plus proche de 0
`round(5.6)` $\rightsquigarrow 6$ `round(-5.6)` $\rightsquigarrow -6$	de l'encadrement à l'unité près de l'argument, cette commande retourne l'entier le plus éloigné de 0
`float(sqrt(2))`	retourne une valeur approchée d'un nombre avec maximum 15 chiffres dans la partie décimale. On utilisera la fonction `bfloat()` pour passer cette limitation.
`bfloat()`	La commande `bfloat()` affiche la valeur décimale approchée du nombre passé en argument
`fpprec`	Le paramètre `fpprec` définit le nombre de chiffres utilisés dans la partie décimale par la commande `bfloat()`.

2. *Les nombres complexes :*

`realpart(1+%i)`	retourne la partie réelle d'un nombre
`imagpart(1+%i)`	retourne la partie imaginaire d'un nombre ;
`cabs(1+%i)`	affiche le module d'un nombre complexe
`carg(1+%i)`	affiche l'argument d'un nombre complexe
`rectform(e^(%i*2*%pi/3))`	retourne la forme algébrique du nombre complexe
`polarform(1+%i)`	retourne la forme exponentielle du nombre complexe
`conjugate(1+%i)`	retourne le conjugé du nombre complexe

3. *Calculs formels :*

`expand((x+1)*(x^2-1))`	développe une expression
`factor(x^2+2*x+1)`	factorise une expression
`divide(x^3+x^2-1,x+1)`	effectue la division d'un polynome par un autre et retourne une liste de deux éléments : le quotient et le reste
`partfrac((x+1)/(x-1),x)`	affiche les éléments simples composant une fraction rationnelle
`num((x+1)/(x^2))`	affiche le numérateur d'un quotient
`denom((x+1)/(x^2))`	affiche le dénominateur d'un quotient
`rat(...)`	affiche un polynôme ou une fraction rationnelle sous forme développée réduite en fonction de l'indéterminé passée en second argument : `rat(expand((2*x+1)*(a*x^2+b*x+c)),x)` $\rightsquigarrow$ `2*a*x^3+(2*b+a)*x^2+(2*c+b)*x+c`
`ratcoef(2*x^3+2*x-1,x,3)`	retourne le coefficient du monôme en "x" de degré 3.
`diff(x^2+x-1,x)`	dérive une expression en fonction de la variable passée en second argument

`integrate(x+1,x)`	déterminer la primitive d'une expression en précisant la variable d'intégration. En fournissant, en arguments supplémentaires, les bornes d'intégration, Maxima retourne la valeur de l'intégrale : `integrate(x+1,x,1,6)`

`solve(x^2-4*x+1)`	détermine les racines de ce polynôme ou des équations simples :
	`solve(3*%e^x-1=3)`
	Elle permet également de résoudre un système d'équations linéaires :
	`solve([x+y-1=0,2*x-y+1=0],[x,y])`
	Voici deux valeurs particulières retournées par cette commande :
	➪ `solve(0*x=0)` : retourne `all` ➪ `solve(0*x=1)` : retourne `[]`

`ev(x^2+1,x=2)`	évalue l'expression pour `x=2`. Cette commande peut également être utilisée pour effectuer une substitution dans une expression : `ev(5*x^2-2*x+1,x=y+1)` $\rightsquigarrow$ $5(y+1)^2-2(y+1)+1$
`taylor(log(x),x,1,4)`	développement de Taylor pour x_0=1 à l'ordre 4.

4. *Probabilité :* *(il faut charger le module* `load(distrib)` *)*

`binomial(10,2)`	retourne le coefficient binomial $\binom{10}{2}$
`pdf_binomial(2,3,0.3)`	retourne la valeur de $\mathcal{P}(\mathcal{X}=2)$ pour $\mathcal{X}\sim\mathcal{B}(3;0{,}3)$
`cdf_binomial(2,3,0.3)`	retourne la valeur de $\mathcal{P}(\mathcal{X}\leqslant 2)$ pour $\mathcal{X}\sim\mathcal{B}(3;0{,}3)$
`quantile_binomial(2,3,0.3)`	retourne la valeur de α réalisant $\mathcal{P}(\mathcal{X}\leqslant\alpha)=2$ pour $\mathcal{X}\sim\mathcal{B}(3;0{,}3)$
`cdf_continuous_uniform(3.5,3,4)`	retourne la valeur de $\mathcal{P}(\mathcal{X}\leqslant 3{,}5)$ pour $\mathcal{X}\sim\mathcal{U}([3;4])$
`quantile_continous_uniform(0.1,3,4)`	retourne la valeur de α réalisant $\mathcal{P}(\mathcal{X}\leqslant\alpha)=0{,}1$ pour $\mathcal{X}\sim\mathcal{U}([3;4])$
`cdf_normal(2,3,4)`	retourne la valeur de $\mathcal{P}(\mathcal{X}\leqslant 2)$ pour $\mathcal{X}\sim\mathcal{N}(3;4^2)$
`quantile_normal(0.1,3,4)`	retourne la valeur de α réalisant $\mathcal{P}(\mathcal{X}\leqslant\alpha)=0{,}1$ pour $\mathcal{X}\sim\mathcal{N}(3;4^2)$

5. *Calculs matriciels :*

`matrix([a,b,c],[c,d,e])`	définit la matrice $\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix}$
`A.B`	La multiplication entre deux matrices. L'opérateur est le point de la ponctuation.
`A^^n`	la puissance d'une matrice. Attention, il faut doubler le symbole `"^"`
`determinant(A)`	retourne le déterminant de la matrice `A`
`eigenvectors(A)` `eigenvalues(A)`	retourne les vecteurs propres et les valeurs propres de la matrice `A`
`invert(A)`	retourne la matrice inverse de la matrice `A`
`transpose(A)`	retourne la matrice transposée de la matrice `A`

6. *Commande pour la programmation :*

`is(3>%pi)`	retourne `false`. La commande `is()` permet d'effectuer un test
`not(is(3>%pi))`	retourne `true` ; le résultat est l'opposé du test `is(3>%pi)`
`freeof(x^2+y*x,z)`	vérifie si la variable du deuxième argument est présente dans l'expression du premier argument. Cet exemple retourne `true`
`numberp(m)`	retourne `true` si `m` est un nombre numérique
`emptyp(m)`	retourne `true` si `m` est une liste vide
`display(x,y)`	cette commande permet de forcer l'affichage de résultats *(ici, les valeurs de* `x` *et* `y`*)*, notamment lors de l'exécution de boucles
`printf()`	Considérons la commande : `x:2.2;  printf(false,"~d ~a ~h",x,x/3,x/3);` cette commande permet d'écrire les résultats de Maxima dans des fichiers, mais ici avec l'argument `false`, les résultats s'afficheront à l'écran. L'appel à cette commande affichera la valeur de `x` sous la forme d'un entier *(*`~d`*)* et deux fois la valeur de `x/3` respectivement avec la forme d'affichage par défaut de Maxima *(*`~a`*)*, puis sous la forme d'un grand nombre en virgule flottante *(*`~h`*)*
`kill(x)`	permet d'effacer l'affectation de la variable `x` dans la session courante de Maxima. La commande `kill(all)` permet d'effacer toutes les affectations de la session courante.