

MÉMO MAPLE

Notions de base

On appelle prompt le signe en début de ligne: **>**
Pour qu'une ligne de commande soit exécutée et le résultat affiché, il faut que cette ligne se termine par un point virgule (;). Si on place deux points (:) à la fin de la ligne, le résultat n'est pas affiché.

Pour accéder à la page d'aide de *fonction*, on entre après le prompt : **?fonction**. On peut également placer le curseur sur **fonction** et taper  + .

On affecte une variable à l'aide de **:=** :

>var:=valeur :

Le quote (`) bloque l'évaluation et permet de désaffecter une variable:

>var := 'var' :

Le double quote ou dito (") permet de rappeler le résultat précédent.

La fonction **restart** démarre une nouvelle session Maple.

La fonction **with** permet de charger un package:
>with(package) :
Les principaux packages sont:

Nom	Description
combinat	Analyse combinatoire
DEtools	Outils pour les équations différentielles
linalg	Algèbre linéaire (notamment les matrices)
numtheory	Manipulation des nombres
plots	Offre de nouvelles fonctions graphiques

Fonctions et Constantes Mathématiques

Notation	Description	Exemple
+ , - , * , / , ^ ou **	Somme, Soustraction, Multiplication, Division et puissance	>3*x^2+2*x-6:
sin , cos , tan , arcsin , arccos , arctan	Fonctions trigonométriques circulaires	>tan(theta) :
sinh , cosh , tanh , arcsinh , arccosh , arctanh	Fonctions trigonométriques hyperboliques	>sinh(0) :
exp	Fonction exponentielle	>exp(x) :
ln ou log	Logarithme népérien	>ln(a) :
log[b]	Logarithme en base <i>b</i> . En particulier, log[10] renvoie le logarithme décimal	>log[10](a) :
abs	Renvoie la valeur absolue d'un nombre réel ou le module d'un nombre complexe	>abs(-5) : >abs(1+I) :
sqrt	Fonction racine carrée	>sqrt(x) :
!	Factorielle	>n!
Pi	La valeur de π	
I	Le <i>i</i> complexe (tel que $i^2 = -1$)	
infinity	L'infini (utile pour les limites)	

Maple permet également de créer ses propres fonctions avec **> :**
>f:=(x,y)->cos(x*y)+x^2*y^2:
On calcule alors la valeur de la fonction par:
>f(2,3) ;

$\cos(6) + 36$

Fonctions et commandes

Notation	Description	Exemple
abs	Valeur absolue ou module d'un nombre complexe	>abs(-3) : abs(1+I) :
animate	Crée une animation en faisant varier un paramètre (package plots)	>animate(cos(x-a), x=-5..5, a=0..5) :
binomial	Renvoie les C_p^n	>binomial(5,3) :
ceil	Entier immédiatement supérieur à <i>n</i>	>ceil(n) :
collect	Rassemble les termes de même degré dans une expression polynomiale	>collect((x+1)^3+(x-1)^2),x) :
combine	Linéarise une expression polynomiale	>combine(cos(x)^3) :
convert	Permet de convertir un type d'objet Maple en un autre	>convert((x^2+1)/(x+1), parfrac,x) :
D	Dérive une fonction	>D(sin) :
det	Calcul du déterminant d'une matrice (package linalg)	>det(M) :
diff	Calcule la dérivée de l'expression	>diff(sin(x),x) :
display	Affiche deux graphiques simultanément (package plots)	>a:=plot(sin) : >b:=plot(cos) : >display(a,b) :
dsolve	Résout une équation différentielle	>dsolve(diff(y(x),x)+2*y(x)=x,y(x)) :
evalf	Renvoie une approximation numérique du résultat	>evalf(exp(1)) :
expand	Développe une expression	>expand((x+1)^2) :
evalm	Affiche une matrice	>evalm(M) :
factor	Factorise une expression polynomiale	>factor(x^2-2*x+1) :
floor	Prend la partie entière d'un nombre	>floor(19.02) :
ifactor	Décompose un entier en produit de nombres premiers	>ifactor(4348) :
igcd	Plus grand commun diviseur	>igcd(3,15) :
ilcm	Le plus commun multiple	>ilcm(3,5) :
Im	Renvoie la partie imaginaire d'un nombre complexe	>Im(1+2*I) :
int	Permet de trouver la primitive / l'intégrale d'une expression (si on place deux bornes)	>int(x,x) : >int(x, x=a..b) :
inverse	Calcule l'inverse de la matrice <i>M</i> (package linalg)	>inverse(M) :
iquo	Quotient de la division euclidienne de <i>a</i> par <i>b</i>	>iquo(a,b) :

Notation	Description	Exemple
irem	Reste de la division euclidienne de a par b	>irem(a,b) :
isprime	Teste si n est premier	>isprime(n) :
lhs	Isolé le membre de gauche d'une équation	>lhs(2*x=1-x) :
limit	Calcule la limite d'une expression	>limit(1/x, x=0, left) :
map	Applique une fonction aux éléments d'une liste	>map(ln, [a,b,c,d]) :
member	Teste l'appartenance d'un opérande à une liste	>member(a, [a,b,c,d]) :
nops	Compte les opérandes d'une expression	>nops(L) :
normal	Réduit l'expression au même dénominateur	>normal(1+1/x) :
plot	Permet de créer un graphique	>plot(sin(x), x=-pi..pi) :
print	Affiche une expression sur une feuille de calcul	>print(Maple) :
rand	Génère aléatoirement un entier	>rand() :
rank	Calcule le rang de la matrice M (package linalg)	>rank(M) :
Re	Renvoie la partie réelle d'un nombre complexe	>Re(1+2*I) :
rhs	Isolé le membre de droite d'une équation	>rhs(2*x=1-x) :
rsolve	Permet de trouver une récurrence	>rsolve({u(n+1)=2*u(n)+1, u(0)=1}, u(n)) :
seq	Génère une séquence	>seq(i^2, i=1..5) :
simplify	Simplifie une expression	>simplify(sin(x)^2+cos(x)^2) :
solve	Résout une équation	>solve(x^2-x+1=0, x) :
subs	Substitue un opérande à un autre	>subs(3=4, 3*x) :
sort	Permet de trier une expression ou une structure	>sort(expression) :
sum	Réalise une somme	>sum(1/k, k=1..5) :
taylor	Effectue un développement limité	>taylor(sin(x), x=0, 5) :
type	Permet de tester le type d'objet Maple de l'expression	>type(2, integer) :
whattype	Renvoie le type d'objet Maple de l'expression	>whattype(5) :

N.B. On a fait figurer entre parenthèses les éventuels packages auxquels une fonction appartient. Ce package doit alors être chargé pour que la fonction soit utilisable.

Graphisme

La fonction **plot** permet de faire un graphique :

```
>plot(expression(x),
x=min..max);
>plot(expression(theta), theta=min..max,
coords=polar);
```

Une courbe paramétrée est obtenue par :

```
>plot([x(t),y(t),t=min..max]);
```

Une courbe en coordonnées polaires s'obtient par :

```
>plot(expression(theta), theta=min..max,
coords=polar);
```

Les graphiques en trois dimensions sont obtenus avec la fonction **plot3d** :

```
>plot3d(expression(x,y),
x=min1..max1, y=min2..max2);
```

Séquences, Listes, Ensembles et Tableaux

Une séquence est une succession d'opérandes séparés par des virgules : a, b, c, d . Une liste est une séquence entourée de crochets : $[a, b, c, d]$. Un ensemble est une séquence encadrée par des accolades. Considérons la liste L :

```
>L:=[a,b,c,d];
```

On peut extraire le $k^{\text{ème}}$ opérande avec la fonction **op** ou directement par **L[k]** :

```
>op(2,L), L[2];
```

b, b

On peut compter le nombre d'opérandes de L :

```
>nops(L) ;
```

4

map permet d'appliquer une fonction à tous les éléments d'une liste :

```
>map(cos, L) ;
```

$[\cos(a), \cos(b), \cos(c), \cos(d)]$

On définit un tableau contenant les valeurs $v_{1,1} \dots v_{m,n}$ avec la fonction **array** :

```
>T:=array([[v1,1,...,v1,n], ..., [vm,1,...,vm,n]]);
```

On peut affecter une cellule de ce tableau :

```
>T[i,j]:=v4,j;
```

L'opérateur **\$** est pratique pour générer des séquences :

```
>k $k=1..3;
```

1,2,3

Pour créer une séquence vide on lui affecte la valeur **NULL**

Procédures

Une procédure doit être entrée dans un même bloc d'instructions. Pour cela, pour revenir à la ligne, il faut taper  +  (et non pas simplement ).

On définit une procédure par :

```
>nom :=proc(arg1,...argn)
local v1,...vn;
global vn+1,...vnm;
...
end;
```

Les variables déclarées globales sont affectées en sortie de la procédure. En revanche, les variables locales ne sont pas affectées en sortie de procédure.

Boucle **for** :

```
>for var from ini to fin do
instructions;
od ;
```

On peut également sortir d'une boucle **for** à l'aide de la commande **break**

Boucle **while** :

```
>while condition do
instructions;
od;
```

Boucle **if** :

```
>if condition1 then
instructions1;
elif condition2 then
instructions2;
...
else
instructionsn;
fi;
```