

Université Mohammed V - Agdal  
Faculté des Sciences  
Département de Mathématiques et Informatique  
Avenue Ibn Batouta, B.P : 1014  
Rabat, Maroc

**Filière :**  
**Sciences Mathématiques et Informatique (SMI)**  
**Sciences Mathématiques (SM)**  
**&**  
**Sciences Matière Physiques (SMP)**

# **MAPLE**

## **Une Introduction**

Par

**EL GHAZI A. & EL HAJJI S.**

[aelghazi@msn.com](mailto:aelghazi@msn.com)    [elhajji@fsr.ac.ma](mailto:elhajji@fsr.ac.ma)

**Groupe Analyse Numérique et Optimisation**

<http://www.fsr.ac.ma/ANO>

**Année 2003-2004**

# Table des matières

I.	Introduction .....	3
II.	Ce que Maple peut faire pour vous .....	4
1.	Arithmétique et Calculs numériques .....	4
2.	Fonctions particulières .....	6
3.	Calculs polynomiaux et rationnels .....	7
4.	Calculs trigonométriques.....	7
5.	Calcul de dérivées .....	8
6.	Développements limités .....	8
7.	Intégration .....	8
8.	Ensembles, listes .....	9
9.	Graphiques .....	10
10.	Algèbre linéaire .....	11
III.	Programmation en Maple .....	12
1.	Structures de contrôle.....	12
2.	Flèches et procédures .....	14
IV.	Annexe : tableaux récapitulatifs.....	18

# I. Introduction

Le calcul formel est un domaine charnière entre les mathématiques et l'informatique dont l'objectif est d'obtenir des algorithmes efficaces pour manipuler les objets formels usuels des mathématiques tels que les fonctions usuelles, les fractions rationnelles, les matrices, les polynômes etc...

Maple est un système de calcul formel produit par WMSG (Waterloo Maple Software Group). Il peut aussi être considéré comme un logiciel permettant de faire :

Calcul numérique

Calcul symbolique

Ces calculs peuvent être faits

Sous forme Interactive : mode commande

ou exécuter sous forme programme (procédure).

On utilisera Maple pour sa fonction première et aussi en tant qu'atelier permettant de faire de l'algorithmique mathématique.

## Structure de Maple

Maple comme tout système de calcul formel offre deux fonctions :

- L'interpréteur : c'est la fonction "super calculatrice" du système. En mode interactif, l'utilisateur entre les instructions et le système effectue sur le champ les calculs demandés.
- Le langage de programmation : l'utilisateur définit ses propres modules, les primitives du système deviennent alors des procédures de haut niveau que l'on peut utiliser.

Ces deux aspects sont complémentaires.

## Le système d'aide intégré

Maple comporte une fonction d'aide intégrée qui rappelle la totalité des fonctions et des commandes de Maple. On peut se reporter souvent à cette aide pour y obtenir de l'aide sur la syntaxe des commandes et y puiser de précieux exemples et idées.

Le "Help" peut être utilisé de plusieurs façons :

- navigation : chaque page d'aide comporte des liens hypertextes sur lesquels on peut cliquer pour obtenir des informations supplémentaires, dans le menu "Help" sélectionnez "contents" puis naviguez entre les différentes pages à l'aide des liens.
- recherche d'une fonction précise : pour obtenir de l'aide sur une fonction comme **solve** dans le menu "Help", on clique sur "Topic Search", dans la zone "Topic" on entre le nom de la fonction sur laquelle on désire obtenir de l'aide, on active le bouton "Apply" pour lancer la recherche sans fermer la zone de dialogue, ou on active le bouton "OK" qui relance la recherche et ferme la zone de dialogue.
- recherche d'un mot ou d'une phrase quelconque : ce système d'aide permet de rechercher automatiquement dans l'ensemble des pages d'aide celles qui contiennent une chaîne de caractères donnée. Dans le menu "Help", on sélectionne "Full Text Search", dans la zone "Word(s)" on entre le mot ou la phrase sur lequel on désire obtenir de l'aide, la suite des opérations est similaire à ce qui a été vu précédemment.

## Fenêtre Maple

### Le haut de la fenêtre

Le haut de la fenêtre est composé de trois barres :

- **La barre de menu** : on y trouve les commandes suivantes :

- File qui gère les feuilles de travail (ouverture, création, sauvegarde d'un fichier....)
- Edit qui permet entre autre de sélectionner, copier, coller, couper, annuler une action, exécuter une feuille de travail.....
- View qui donne accès aux palettes, permet aussi de fusionner les sections, de zoomer.....
- Insert qui permet de créer les paragraphes, les sections, les sous-sections....
- Format qui offre différentes options pour la manipulation du texte, en particulier la commande Convert to permet de convertir du texte simple (respectant la syntaxe Maple) en formule mathématique
- Option.
- Window permet de manipuler les fenêtres, en particulier avoir à sa disposition une fenêtre brouillon et une fenêtre propre via la commande Vertical
- Help pour l'aide.

- **La barre d'outils**

- les quatre premiers boutons à partir de la gauche permettent la manipulation des fichiers
- les trois suivants servent de presse-papiers
- [ $\>$ ] sert à insérer une ligne de commande
- le bouton STOP devient rouge dès qu'une commande est lancée, essentiel pour arrêter un calcul qui s'éternise!!!!

- **La barre de contexte**

### Le bas de la fenêtre

Le bas de la fenêtre contient une barre d'état qui nous informe sur le temps de calcul CPU et sur la place mémoire utilisée.

### Le centre de la fenêtre

C'est dans cette partie de la fenêtre que va se dérouler la feuille de travail où le texte est en noir, les instructions pour la machine en rouge et les résultats calculés par la machine en bleu.

## II. Ce que Maple peut faire pour vous

### 1. Arithmétique et Calculs numériques :

Maple fait de l'arithmétique et connaît un très grand nombre de fonctions, en plus de faire du calcul différentiel et intégral, de l'algèbre linéaire, etc... Nous faisons ici « une » brève présentation de ces quelques aspects de ce langage.

**Constantes.** Il est capable de reconnaître des constantes de différents types :

entier : **5, 2**,...  
fraction : **1/5, 11/10**, ...  
décimal : **2.333, 3.65**, ...  
symbolique : **Pi, I, E, infinity** ...

**Nombres différents.** Pour Maple les nombres **1/5** et **0.2** ne sont pas identiques. Cette distinction permettra, d'écrire exactement **1/3** et d'effectuer des calculs sans aucune erreur d'arrondi sur des entiers et des fractions. Pour effectuer l'évaluation numérique, il suffit d'employer la commande **evalf**

**Arithmétique** Maple connaît évidemment les opérations arithmétiques.

Il interprète les **+, -, /, \*, ^(puissance), !, sqrt (racine carrée)**.

**NB : Maple fait la différence entre MAJUSCULE et minuscule**

**restart.** Réinitialisation de la session Maple.

**%** a pour valeur le dernier résultat, **%%** l'avant-dernier....

### Exemple :

Si on connaît la commande Maple, il suffit de l'exécuter. Sinon on peut demander sa syntaxe en allant dans le help ou tout simplement en l'exécutant précédé du signe ?.

```
> restart ;
```

```
> ?evalf ;
```

```
> Pi; # un commentaire sur la ligne
```

Remarque : Tout ce qui vient après # est un commentaire

$\pi$

```
> evalf(Pi); #par défaut, la valeur approchée de Pi est donnée avec 10  
chiffres significatifs
```

3.141592654

```
> evalf(Pi, 8); #valeur approchée de Pi avec 8 chiffres significatifs
```

3.1415927

```
> 5+3;
```

8

```
> 2*4.5;
```

9.0

```
> 4/5;
```

$\frac{4}{5}$

```
> evalf(4/5);
```

0.8000000000

> **25!**;

>

15511210043330985984000000

> **2^3**;

8

> **sqrt(sqrt(5)+2)**;

$\sqrt{\sqrt{5} + 2}$

Si on ne connaît pas le nom, alors il faut se « rappeler que Maple » est rédigé en **anglais** et donc trouver le terme adéquat en anglais (*grosse difficulté pour les débutants*). Il faut s'aider avec les exemples de Maple ou utiliser un index qui fournit les 100 commandes les plus usuelles. Par exemple si on veut résoudre une équation on lance le help avec le mot anglais solve qui veut dire résoudre :

> **?solve**;

> **solve(x^2-3\*x+2,x)**;

2, 1

## 2. Fonctions particulières

- **Sum, sum** fait la somme des termes :

> **Sum(k,k=1..10)**;

$\sum_{k=1}^{10} k$

> **sum(k,k=1..10)**;

55

- **Product, product** fait le produit des termes

> **Product(k, k=1..10)**;

$\prod_{k=1}^{10} k$

> **product(k, k=1..10)**;

3628800

- **ifactor**, décomposition en facteur premier.

> **ifactor(120)**;

$(2)^3 (3) (5)$

NB : **factor** permet de factoriser des polynômes et non des entiers

- **abs**, la valeur absolue.

> **abs(-5);**

5

- **iquo(a,b)**, pour le quotient de la division entière de a par b.

> **iquo(20,3);**

6

- **irem(a,b)**, pour le reste de la division entière de a par b.

> **irem(20,3);**

2

- **igcd(a,b,...,k)**, pour le pgcd des entiers a,b,...k.

> **igcd(100,20,45);**

5

- **ilcm(a,b,...,k)**, pour le ppcm des entiers a,b,...k

> **ilcm(10,15,4,12);**

60

### 3. Calculs polynomiaux et rationnels

- Développements et simplification de polynômes.

> **expand((x-2)^5);**

$$x^5 - 10x^4 + 40x^3 - 80x^2 + 80x - 32$$

> **P:=x^8+x^4+1;**

$$P := x^8 + x^4 + 1$$

> **factor(P);**

$$(x^2 + x + 1)(x^2 - x + 1)(x^4 - x^2 + 1)$$

- Décomposition de fractions rationnelles en éléments simples.

> **f:=7/((x+1)^7-x^7-1);**

$$f := \frac{7}{(x+1)^7 - x^7 - 1}$$

> **convert(f,parfrac,x);**

$$-\frac{1}{(x^2+x+1)^2} - \frac{1}{x+1} + \frac{1}{x} - \frac{1}{x^2+x+1}$$

### 4. Calculs trigonométriques

- Linéarisation des expressions trigonométriques.

> **s:=sin(x)^5;**

$$s := \sin(x)^5$$

> **combine(s, trig);**

$$\frac{1}{16} \sin(5x) - \frac{5}{16} \sin(3x) + \frac{5}{8} \sin(x)$$

- Expression de lignes trigonométriques

> **expand(cos(5\*x));**

$$16 \cos(x)^5 - 20 \cos(x)^3 + 5 \cos(x)$$

## 5. Calcul de dérivées

- Dérivée d'une fonction.

> **diff(arctan(x), x);**

$$\frac{1}{1+x^2}$$

- Dérivée cinquième d'une fonction.

> **diff(arctan(x), x\$5);**

$$\frac{384x^4}{(1+x^2)^5} - \frac{288x^2}{(1+x^2)^4} + \frac{24}{(1+x^2)^3}$$

## 6. Développements limités

- **series(g, x, n)**, calcul de le développement limités de la fonction g a l'ordre n.

> **g:=sin(x);**

$$g := \sin(x)$$

> **series(g, x, 6);**

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 + O(x^7)$$

## 7. Intégration

- Calcul de primitives.

> **f:=x\*ln(x);**

$$f := x \ln(x)$$

> **int(f, x);**

$$\frac{1}{2}x^2 \ln(x) - \frac{x^2}{4}$$

- Calcul d'intégrales définies.

> **Int(f, x=1..2);**

> **int(f, x=1..2);**

$$2 \ln(2) - \frac{3}{4}$$

## 8. Ensembles, listes

Un ensemble (type set) est non ordonné : {suite d'expressions}

Une liste (type list) est ordonnée : [suite d'expressions]

{ } est l'ensemble vide et [] est la liste vide.

Le tableau 1 résume les principales différences entre une suite d'expressions, une liste et un ensemble.

Tableau 1

Objet	Caractéristiques
<b>Liste</b>	<b>type : list</b>
	syntaxe : [a,b,c] ordonné : [a,b,c] ≠ [b,a,c] éléments répétés : [a,a,c] → [a,a,c] plusieurs niveaux : l := [b,c]; [a,l,d] → [a,[b,c],d] liste vide : []
<b>Ensemble</b>	<b>type : set</b>
	syntaxe : {a,b,c} non ordonné : {a,b,c} = {b,a,c} pas d'éléments répétés : {a,a,c} → {a,c} plusieurs niveaux : l := {b,c}; {a,l,d} → {a,{b,c},d} ensemble vide : {}
<b>Suite d'expressions</b>	<b>type : exprseq</b>
	syntaxe : a,b,c ordonné : a,b,c ≠ b,a,c éléments répétés : a,a,c → a,a,c un seul niveau : l := b,c; a,l,d → a,b,c,d suite vide : NULL

> **l := [12, -3, 4];**

l := [12, -3, 4]

> **l[2];**

-3

> **s := 1, 2, 4;**

s := 1, 2, 4

> **m := [s, 7];**

m := [1, 2, 4, 7]

> **d := {1, 2, 4, 7, 7};**

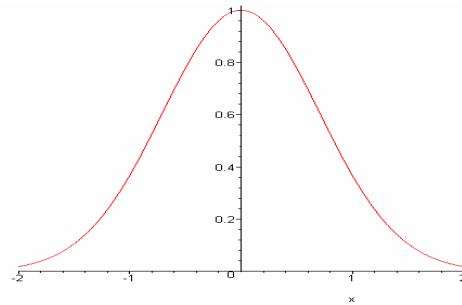
d := {1, 2, 4, 7}

## 9. Graphiques

Les possibilités graphiques de Maple sont assez larges, et différentes options sont utilisables suivant les "packages" utilisés.

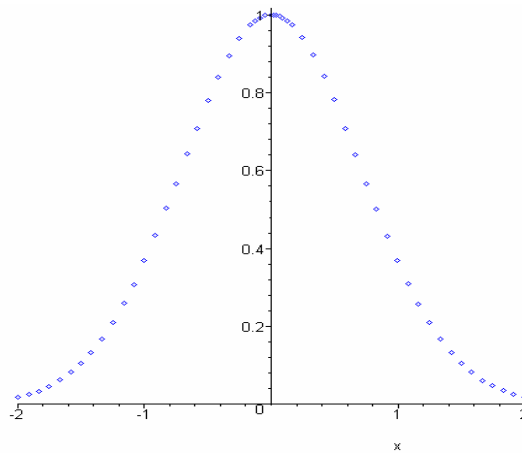
- Représentation des fonctions à une variable.

```
> plot(exp(-x^2),x=-2..2);
```



- Tracer avec des points et changer de couleur.

```
> plot(exp(-x^2),x=-2..2,style=point,color=blue);
```



- ```
plot(exp(-x^2),x=-2..2,style=point,symbol=circle,color=blue);
```

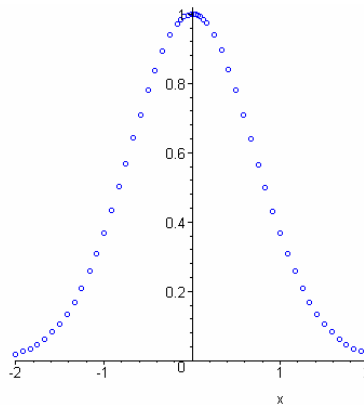


Figure 1

- Représenter plusieurs graphes sur le même dessin

```
> plot({sin(x),cos(x)},x=-Pi..Pi);
```

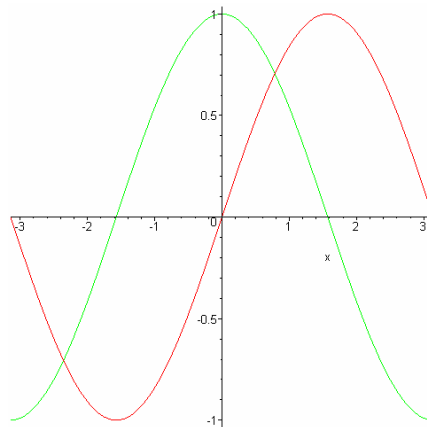
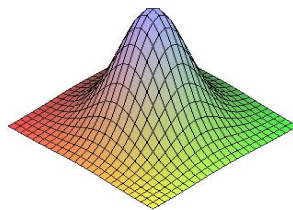


Figure 2

- Surfaces

```
> plot3d(exp(-(x^2+y^2)),x=-2..2,y=-2..2);
```



## 10. Algèbre linéaire

Pour manipuler matrices (matrix) et vecteurs (vector), il existe le package linalg dans lequel on trouve un grand nombre de fonctions pour les opérations sur les matrices et les vecteurs. La fonction vector permet de créer des vecteurs. Ces vecteurs sont considérés dans les calculs comme des vecteurs colonnes, même si cela n'apparaît pas à l'impression.

Pour transformer un vecteur en matrice, on utilise la fonction convert.

```
> v:=vector([1,7,3]);
```

```
v := [1, 7, 3]
```

```
> convert(v,matrix);
```

```
⎡ 1 ⎤  
⎢ 7 ⎥  
⎣ 3 ⎦
```

La fonction matrix permet de créer facilement des matrices. Elle admet comme argument optionnel une fonction qui permet de créer les éléments de la matrice.

```
> B:=matrix(3,2,[1,2,3,4,5,7]);
```



Les structures conditionnelles ne permettent que la distinction de deux cas complémentaires, la syntaxe complète :

```
if condition 1 then action 1
  elif condition 2 then action 2
  elif condition 3 then action 3
  .....
  elif condition N then action N
  else action N+1 fi;
```

Dans d'autres situations, il faut pouvoir répéter une action un nombre de fois. Si le nombre de répétitions est connu d'avance, une telle structure se construit par :

```
for compteur from debut to fin by pas condition do action od;
```

- Compteur doit être un identificateur de variable
- Pas est optionnel et peut prendre des valeurs négatives si on veut décroître le compteur de la boucle

```
> for i from 6 by 2 to 10 do print(i) end do;
```

6

8

10

```
> for i from 20 by -3 to 10 do print(i) end do;
```

20

17

14

11

```
> for i from 7 to 10 do print(i) end do;
```

7

8

9

10

Si le nombre de répétitions n'est pas connu d'avance, on a recours à :

```
while condition do action od;
```

```
> while a < 15 do
```

```
  a:=a+1;
```

```
od;
```

a := 11

a := 12

a := 13

a := 14

$$a := 15$$

### 3. Flèches et procédures

- **La flèche**

On peut créer très facilement des fonctions en Maple de la manière suivante :

**f:=(x1,x2,...,xn)-> expression(x1,x2,...,xn);**

ce qui signifie littéralement :

On définit f comme la fonction qui aux variables (x1, x2, ..., xn) associe l'expression (x1, x2, ..., xn)".

#### Exemples

> **f:=x->exp(-x^2);**

$$f := x \rightarrow e^{(-x^2)}$$

> **f(0);**

$$1$$

> **diff(f(x),x);**

$$-2x e^{(-x^2)}$$

Ici x est une variable muette. En effet on a

> **f(y);f(xi);**

$$e^{(-y^2)}$$

$$e^{(-\xi^2)}$$

On peut construire une fonction en utilisant des structures de contrôle :

> **restart;**

> **f:=x-> if x > 0 then ln(x) else 0 fi;**

*f := proc(x) option operator, arrow; if 0 < x then ln(x) else 0 end if end proc*

> **f(2);**

$$\ln(2)$$

> **f(-2);**

$$0$$

On peut transformer une expression en fonction : l'instruction **unapply**

> **restart;**

> **f:=exp(-x^2);**

$$f := e^{(-x^2)}$$

Peut-on calculer la valeur de f en 0 ?

```
> f(0);
```

$$(e^{(-x^2)})(0)$$

Essayons plutôt ceci :

```
> g:=unapply(f,x);
```

$$g := x \rightarrow e^{(-x^2)}$$

```
> g(0);
```

1

- **La procédure**

Une procédure est un ensemble d'instructions ordonnées.

Un programme est une procédure qui en appelle d'autres. Une procédure se construit selon la syntaxe :

```
proc(données);  
action;  
RETURN(résultats);  
end;
```

où **proc**, **RETURN**, **end** sont des mots réservés du langage, tandis que données, action, résultats sont à particulariser.

Pour changer de ligne sans changer de section dans une feuille de calcul, appuyer simultanément sur "Shift" et "Return". Chaque ligne de commande, sauf la ligne comportant "**proc**" doit se terminer soit par ";" soit par ":"

### Exemples

```
> restart;  
> maxi :=proc(a,b)  
print("bonjour");  
if (a > b) then print("le max est a="),a  
                else print("le max est b="), b  
end fi  
end ;  
> maxi(1/2,5/9);
```

"bonjour"

"le max est b="

$\frac{5}{9}$

- **Variables locales et variables globales**

Outre les variables transmises comme paramètres dans une procédure, on distingue deux types de variables :

### a. les variables locales

Elles sont utilisées durant l'exécution de la procédure mais ne sont pas accessibles à l'extérieur de celle-ci.

#### Exemple

```
> restart;
> iter:=proc(n)
local compteur,i;
compteur:=0;
for i from 1 to n do
if irem(n,i)=0 then print(i," est un diviseur de ", n) ;
compteur:=compteur+1;
fi;
od;
print("On compte", compteur," diviseurs de", n);
end;
```

```
iter := proc(n)
local compteur, i;
compteur := 0;
for i to n do
if irem(n, i) = 0 then
print(i, " est un diviseur de ", n); compteur := compteur + 1
end if
end do ;
print("On compte", compteur, " diviseurs de", n)
end proc
```

```
> iter(34);
1, " est un diviseur de ",34
2, " est un diviseur de ",34
17, " est un diviseur de ",34
34, " est un diviseur de ",34
"On compte",4, " diviseurs de",34
```

Et pourtant compteur est inconnu à l'extérieur de la procédure. En effet

```
> compteur;
compteur
```

### **b. les variables globales**

Elles sont utilisées durant l'exécution de la procédure mais sont accessibles à l'extérieur de celle-ci.

#### **Exemple (transformation du précédent)**

Reprenez l'exemple précédent mais déclarez la variable compteur comme variable globale : il suffit à cet effet d'ajouter l'instruction :

**global compteur ;**

Alors si on tape

> **compteur ;**

4

## IV. Annexe : tableaux récapitulatifs

Tableau 1 : affectations et opération élémentaires

| Instruction                                | Description                                          |
|--------------------------------------------|------------------------------------------------------|
| <code>x:=2/3 ;</code>                      | affectation de la valeur rationnelle 2/3 à x         |
| <code>x:='x' ;</code>                      | réinitialisation de x                                |
| <code>P :=3*x^2+4 ;</code>                 | affectation d'un polynôme de $\mathbb{Q}[x]$ à P     |
| <code>a:=sqrt(2) ;</code>                  | affectation de racine carrée de 2 à l'atome a        |
| <code>evalf (a) ;</code>                   | évaluation réelle approchée de a (10 décimales)      |
| <code>Eq1 :=x-2+x+1=0;</code>              | affectation d'une équation à Eq1                     |
| <code>whattype(Eq1);</code>                | type de Eq1                                          |
| <code>with(linalg) ;</code>                | chargement des instructions du package linalg        |
| <code>v :=vector([3,5]);</code>            | affectation du vecteur colonne $(3, 5)^T$            |
| <code>v :=vector (2,t-&gt;x^t) ;</code>    | affectation du vecteur colonne $(x, x^2)^T$          |
| <code>A :=matrix([[x-2,51],[2,1]]);</code> | affectation par ligne de A dans $M_2(\mathbb{R}[X])$ |
| <code>randmatrix(3,3);</code>              | matrice 3 x 3 aléatoire dans $\{-99, \dots, 99\}$    |
| <code>x :=2;map(eval,A);</code>            | évaluation de la matrice A pour $x = 2$              |
| <code>B :=transpose(A);</code>             | transposée de A                                      |
| <code>C=band([-1,3,1],4) ;</code>          | matrice 4 x 4 tridiagonale (3 sur la diagonale)      |
| <code>scalarmul(A,3);</code>               | calcul de 3A (idem avec <code>evalm(3*A)</code> )    |
| <code>multiply(A,B);</code>                | calcul de AB (idem avec <code>evalm(A*B)</code> )    |

Tableau 2 : instructions syntaxiques

| Instruction                                | Description                    |
|--------------------------------------------|--------------------------------|
| <code>Restart</code>                       | Réinitialisation               |
| <code>%</code>                             | expression précédente          |
| <code>if .. then .. else .. fi</code>      | instruction conditionnelle     |
| <code>for from 1 to 10 by 2 do ..od</code> | boucle ( $i = 1, 3, 5, 7, 9$ ) |
| <code>while .. do .. od</code>             | boucle conditionnelle          |

Tableau 3 : Principales opérations sur les polynômes

| Instruction                      | Description                                    |
|----------------------------------|------------------------------------------------|
| <code>P :=6*x*(t+1)^2-6 ;</code> | affectation du polynôme P                      |
| <code>Q :=x*t+1;</code>          | affectation du polynôme Q                      |
| <code>subs(x=2,P) ;</code>       | évaluation de P en $x = 2$                     |
| <code>degree(P,t) ;</code>       | degré partiel de P par rapport à t             |
| <code>lcoeff (P, t) ;</code>     | coefficient dominant de P par rapport à t      |
| <code>expand(P) ;</code>         | développement de P                             |
| <code>sort(P) ;</code>           | mise sous forme ordonnée du polynôme P         |
| <code>rem(P,Q,x);</code>         | reste de la division dans $Q(t)[x]$ de P par Q |
| <code>factor(P) ;</code>         | factorisation de P dans $Q[x, t][t]$           |
| <code>solve(P) ;</code>          | recherche des racines exactes de P             |
| <code>P :=x^4-2;</code>          | nouvelle affectation du polynôme P dans $Q[x]$ |
| <code>fsolve(P);</code>          | solutions approchées de $P = 0$                |