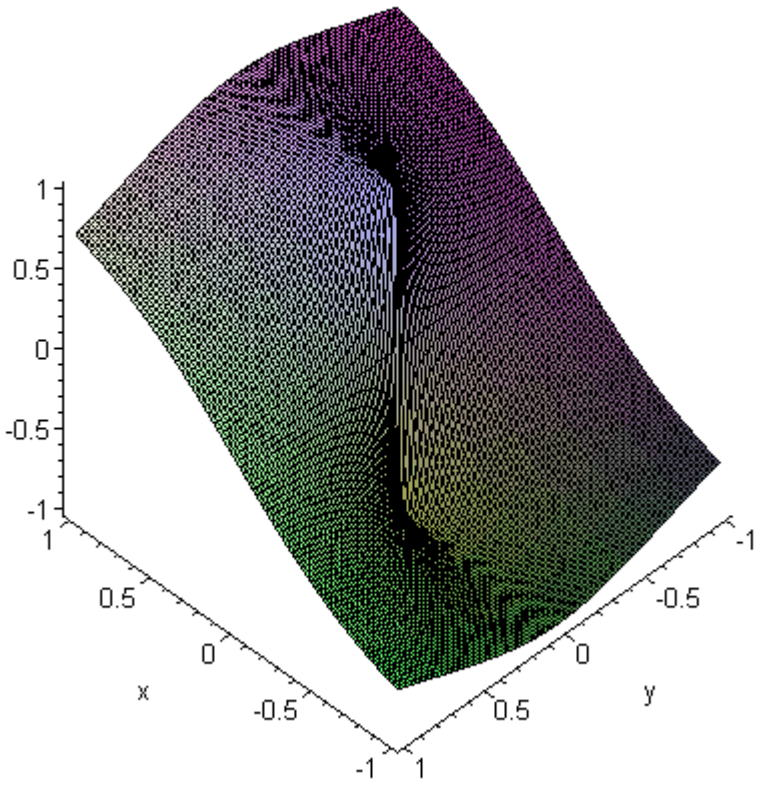
République Algérienne Démocratique et Populaire
Université de Bordj Bou Arreridj
Faculté des Mathématiques et d'Informatique Département d'Informatique

Outils de la programmation mathématique : « Maple »



Par Dr. Abdelouahab Attia

Table des matières

1 Introduction

Généralement un ordinateur est considéré comme un super calculateur, c'est à dire une machine puissante permet de calculer les opérations très compliquées qui ne sont pas faciles à calculer par l'homme, notamment le calcul scientifique. Maintenant, l'ordinateur fait des manipulations très importantes. Il permet de manipuler des symboles, des formules, des équations,...etc. Sur ces symboles. Qui peuvent être utilisés pour des applications très variées dans plusieurs domaines. L'intérêt c'est d'avoir des résultats exacts (c'est à dire sans approximation). Aussi d'arriver à des résultats impossibles par un calcul numérique (par exemple la simplification d'une expression ou d'une fraction, l'expression d'une primitive) mais très utiles pour des applications en physique, mécanique,... Un système de calcul formel (Computer Algebra = Algèbre par ordinateur), est un système capable de traiter des objets mathématiques de manière symbolique.

2 Les outils de calcul formel

Il existe de nombreux outils de programmation pour le calcul scientifique notamment les logiciels de calcul formel (ou symbolique), dont on peut citer:

1. Matlab
2. scilab
3. Gap : http ://www.gap-system.org/
4. Hartmath : http ://www.hartmath.com/
5. Maple : http ://www.maplesoft.com/
6. Mathematica : http ://www.wolfram.com/
7. Mathemagix : http ://www.mathemagix.org/
8. Maxima : http ://www.maxima.fr.st/
9. Mupad : http ://www.mupad.de/

Ces logiciels présentent des différences dans leur utilisation et dans les possibilités qu'ils offrent.

« L'utilisation efficace d'un logiciel de Calcul Formel dépend de manière essentielle des connaissances algorithmiques, et plus généralement mathématiques de l'utilisateur, et exige le même niveau de rigueur que tout autre travail mathématique. »

3 Maple

Maple est un logiciel de calcul formel, souvent désigné comme un outil de calcul symbolique, largement utilisé pour manipuler des expressions mathématiques. Parmi les logiciels de ce type, Maple est l'un des plus puissants et également parmi les plus répandus. Bien qu'il offre des fonctionnalités de calcul numérique, ce n'est pas son objectif principal.

Maple est considéré comme une calculatrice extrêmement puissante en mode programmation. Il offre la possibilité de créer vos propres programmes, procédures, modules, etc. Les procédures que vous créez peuvent être utilisées de la même manière que les fonctions préexistantes de Maple.

4 L'objectif de ce cours

L'objectif de ce cours est de fournir les bases de l'utilisation de Maple, permettant de bien comprendre l'outil, les utiliser et de le maitriser. Dans ce cours, nous montrerons comment utiliser Maple, quelles sont les notations de Maple, comment manipuler des expressions mathématiques, définir des fonctions, résoudre des systèmes linéaires, des équations différentielles et tracer des fonctions en 2 et 3 dimensions. On montrera aussi comment programmer sous Maple. Ces différentes notions seront illustrées par des exemples.

Ce cours peut être considéré comme un tutoriel concis visant à familiariser rapidement les étudiants de première année en licence de mathématiques et informatique avec Maple, en utilisant des exemples.

5 Les premières fonctionnalités du Maple

Maple fonctionne en mode direct (interactif), à l'image d'une calculatrice très puissante, et en mode programmation. Toute fenêtre ouverte sous Maple s'appelle une feuille de calcul (worksheet, en anglais.)

5.1 L'invite de commande, commande, message erreur

Le prompt de Maple est, par défaut, le symbole > après lequel Toutes les commandes passées à l'ordinateur (c'est à dire où la saisie doit être effectuée).

En suite presser la touche Entrée pour transmettre l'ordre au logiciel Maple.

Par exemple, pour calculer $\sqrt{a^2+b^2}$ ou $\sum_{i=1}^{100} i\text{^}2$, on tape puis on valide en appuyant sur la touche Entrée :

```
> sqrt(a^2+b^2);
```

$$\sqrt{a^2+b^2}$$

```
> Sum(i^2,i=1..100);
```

$$\sum_{i=1}^{100} i^2$$

```
> sum(i^2,i=1..100);
```

$$338350$$

Deux possibilités pour les terminaisons des commandes doivent (1) par un point-virgule ; ou (2) par deux points :

```
> sum(i^2,i=1..100):
```

Si la ligne se termine par ' :' points, exécuté la commande mais n'affiche pas le résultat final.

Si la ligne se termine par deux points virgule, exécuté la commande avec affichage du résultat final.

5.2 Système d'aide

Le logiciel Maple est doté d'un système d'aide qui explique les fonctions avec des exemples.

: Pour Obtenir de l'aide sur une fonction en utilisant ? ou help

Exemple

Ici on cherche de l'aide sur la fonction isprime.

```
> ??isprime;
```

```
> help(isprime);
```

Des messages d'erreur peuvent apparaître en cas de mauvaise saisie ou d'opérations illicites :

```
> 12+abs(4-3*sqrt(5);
```

```
Error, `;` unexpected
```

```
> 1/cos(Pi/2);
```

```
Error, numeric exception: division by zero
```

```
> 12+abs(4-3*sqrt(5));
```

5. 3 Les lettres grecques

Pour obtenir les lettres grecques, il suffit de taper leurs noms comme suit :

Lettre	*Saisie*	*Lettre*	*Saisie*	*Lettre*	*Saisie*	*Lettre*	*Saisie*
α	alpha	Α	Alpha	λ	lambda	Λ	Lambda
β	beta	Β	Beta	μ	mu	Μ	Mu
γ	gamma	Γ	Gamma	ν	nu	Ν	Nu
δ	delta	Δ	Delta	ξ	xi	Ξ	Xi
ε	epsilon	Ε	Epsilon	ο	omicron	Ο	Omicron
ζ	zeta	Ζ	Z ETA	π	pi	Π	PI
η	eta	Η	Eta	ρ	rho	Ρ	Rho
θ	theta	Θ	Theta	σ	sigma	Σ	Sigma
ι	iota	Ι	Iota	τ	tau	Τ	Tau
κ	kappa	Κ	Kappa	υ	upsilon	Υ	Upsilon
χ	chi	Χ	CHI	φ	phi	Φ	Phi
ψ	psi	Ψ	Psi	ω	omega	Ω	Omega

5.4 Les Opérateurs

Opérateurs arithmétiques :	Opérateurs d'ordre :	
+ pour l'addition,	< (inférieur à)	> (supérieur à)
− pour la soustraction,	<= (inférieur ou égal à)	
***** pour la multiplication,	>= (supérieur ou égal à)	
/ pour la division (réelle),	= (égal à)	
^ ou ****** pour l'élévation à une puissance,	<> (différent de)	

Opérateurs logiques : not, and, or

Aussi Maple offre d'autre opérateur pour les séquences comme l'opérateur dollar **$** indique une séquence

```
> k $ k=10..15;
```

$$10, 11, 12, 13, 14, 15$$

5.5 Les constantes

Maple offre quelques constantes importantes en mathématique sont reconnues :

Pi π

I imaginaire pur

gamma constante d'Euler

E base des log népériens

± infinity concept traité comme une constante

```
> Pi;- infinity;+ infinity;E; Gamma;I ;
```

$$\pi$$
$$-\infty$$
$$\infty$$
$$E$$
$$\Gamma$$
$$I$$

```
> gamma;
```

$$\gamma$$

```
> evalf(%);
```

0.5772156649

Le caractère "#" permet d'introduire un commentaire dans une feuille de travail.

5.6 LES VARIABLES :

Une ***variable*** est un emplacement mémoire permettant de stocker un objet d'un type donné. Comme exemples : AB, aB, X1, var, x[a] sont des noms de variables MAPLE.

Si l'on affecte à une variable une valeur, la variable est dite ***assignée***.

Si elle n'est associée à aucune valeur, la variable est dite ***non assignée***.

Exemples

```
> X1:=10;
```

$$X1 := 10$$

X1 est une variable assignée, dont la valeur est 10

La variable *a* et la variable indicée x_a sont non assignées :

```
> a,x[a];
```

$$a, x_a$$

Une autre manière d'affecter une valeur à une variable est d'utiliser la fonction ***assign*** :

```
> assign(X1,5);
```

```
> X1;
```

5

Pour désaffecter une variable assignée on appel commande *unassign('var')* permet de vider le contenu d'une variable ' var' comme *restart* qui remet tout à l'état initial.

Exemple :

```
> restart;
```

```
> y:=(x+1)^3-2*x^2-3;
```

$$y := (x+1)^3 - 2x^2 - 3$$

L'expression *y* n'est pas évaluée car la variable x est libre.

```
> x:=2 : y;
```

16

Par contre Z est évalué car *x* n'est pas libre.

```
> z:= x^2+x+1;
```

$$z := 7$$

Si on libère la variable 'x'

```
>unassign('x');# on libère x
>y;z;
```

$$(x+1)^3 - 2\,x^2 - 3$$

$$7$$

On remarque que *z* est toujours constante mais *y* a 'retrouvé la mémoire'.

Vous pouvez désaffecter une variable assignée en écrivant son nom entre deux accents aigus :

```
> z:='z':z;
```

$$z$$

On peut aussi utiliser la fonction evaln (évaluer en un nom) :

```
>  var:=evaln(x):var;
```

$$2$$

6 Les Fonctions prédéfinies

La fonction evalf (**Nombres réels approchés**) :

La fonction **evalf** permet d'évaluer nombre **rationnel ou réel en virgule flottante avec une précision définie. (Variable système par défaut initialisée à 10)**

```
> evalf(Pi);
```

3.141592654

```
> evalf(Pi,15);
```

3.14159265358979

La précision de Maple peut même être modifiée entièrement en utilisant la variable système Digits, qui est une variable globale qui est toujours assignée et est initialisée à 10.

La commande Digits

```
> Digits:=4;
```

$$Digits := 4$$

```
> evalf(Pi);
```

3.142

6 .1 Fonctions circulaires et leurs inverses :

sin(x), **cos(x)**, **tan(x)**

Exemple:

```
> sin(Pi);
```

0

6.2 Fonctions inverses pour les fonctions circulaires :

arcsin(x), **arccos(x)**, **arctan(x)**

Exemple:

```
> arcsin(Pi);
```

$1.570796327 - 1.810991349\,I$

Fonctions hyperboliques :

sinh(x), **cosh(x)**, **tanh(x)**,

exemple:

```
> sinh(Pi);
```

11.53029203

6.3 Fonctions inverses pour les fonctions hyperboliques

arcsinh(x), **arccosh(x)**, **arctanh(x)**

Exemple :

```
> arcsinh(Pi);
```

1.861812557

6.4 Autres fonctions :

Pour tout x, réel,

ln(x) ou **log(x)**	(logarithme népérien de x),
exp(x)	(exponentielle naturelle de x),
log[10](x)	(logarithme décimal de x)
sqrt(x)	(racine carrée de x),
surd(x,n)	(racine nième de x)
abs(x)	(valeur absolue de x)
floor(x)	(partie entière de x)

7 Manipulation des entiers sous Maple

Pour x nombre entier

isprime(x)	Retourne *true si x* est un nombre premier
nextprime(x)	Retourne le nombre premier suivant x

prevprime(x)	Retourne le nombre premier précédant x
ithprime(x)	Retourne le *xème* nombre premier
ifactor(x)	Retourne les facteurs premiers de l'entier x

Exemple :

```
> isprime(17);
```

true

```
> nextprime(17);
```

19

```
> prevprime(17);
```

13

```
> ithprime(5);
```

11

```
> ifactor(120);
```

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

7.1 Autre manipulation : PGCD et PPCM

igcd(x,y,z,...)	Retourne le PGCD de x,y,z, ...
ilcm(x,y,z, ...)	Retourne le PPCM de x,y,z, ...

Exemple

```
> igcd(18,15);
```

3

```
> ilcm(18,15);
```

90

7.2 La division euclidienne

Etant donnés deux entiers relatifs ont et b,
iquo(a,b) (affiche quotient de la division euclidienne de a par b)
irem(a,b) (affiche reste de la division euclidienne de a par b)

Exemple :

```
> iquo(23,5);
```

4

```
> irem(23,5);
```

3

8 Les nombre complexe

Pour z, complexe,

Re(z)	(affiche partie réelle de z)
Im(z)	(affiche partie imaginaire de z)
abs(z)	(permet de calculer le module de z)
argument(z)	(permet de calculer l'argument de z)
conjugate(z)	(donner le nombre conjugué de z)

Exemple

Soit : z=4+5i .

```
> z=4+I*5;
```

$$z = 4 + 5\,I$$

```
> Im(z);
```

$$5$$

```
> abs(z);
```

$$\sqrt{41}$$

```
> argument(z);
```

$$\arctan\left(\frac{5}{4}\right)$$

```
> conjugate(z);
```

$$4 - 5\,I$$

9 Sommes, produits et factorielles

Maple offre des fonctions avec pour le calcul des sommes Σ ou des produits Π dans les expressions mathématiques.

```
> sum(i^2,i=1..5);
```

$$55$$

```
> Sum(i^2,i=1..50);
```

$$\sum_{i=1}^{50} i^2$$

```
> product(i**2,i=1..3);
```

$$36$$

```
> Product(i**2,i=1..50);
```

$$\prod_{i=1}^{50} i^2$$

Lorsque l'instruction est écrite avec une majuscule, il s'agit de la forme inerte (Sum, Product), c'est à dire que l'expression est affichée mais non exécutée.

On obtenir le factoriel d'un nombre à l'aide de la fonction factoriel(n) ou n ! Factoriel.

```
> factorial(n);
```

$$n!$$

```
> factorial(5);
```

$$120$$

```
> 5!;
```

$$120$$

10 Structures mathématiques usuelles

10.1 Séquences Listes, Ensembles,

Une séquence est une suite finie d'éléments de type exprseq, séparés par des virgules :

```
> x:=a,b: x;
```

$$a, b$$

Chaque élément peut être accédé en fonction de son statut, mais il est interdit de l'attribuer.

```
> x:=a,b,c,d,e: x;
```

$$a, b, c, d, e$$

```
> x[3];
```

$$c$$

Une liste, de type list, est obtenue en plaçant une suite finie (x1, ..., xn) (séquence) entre les crochets [et] se note [x1, ..., xn].

Un ensemble fini {x1, ..., xn} se note {x1, ..., xn} et s'appelle un ensemble. En mathématiques, l'ordre des éléments n'est pas important, contrairement à une liste, et lorsqu'un même élément apparaît plusieurs fois, il y a une simplification automatique.

```
> [a,b,c,b,b]; {a,b,c,b,b}; {1+1, 2,4,1+1+1, 1+1};
```

$$[a, b, c, b, b]$$

$$\{a, b, c\}$$

$$\{2, 3, 4\}$$

La liste vide, l''ensemble vide se notent respectivement par les expressions [ ], { }.

```
> Q:={}:Q;
```

$$\{\ \}$$

```
> P:={a,b}: P union Q;
```

$$\{a, b\}$$

```
> [a,NULL]; [NULL];
```

$$[a]$$

$$a, b, c, d, e$$

$$[\ ]$$

NULL désigne la séquence vide. Donc la séquence vide ne provoque aucun affichage :

```
> a,b,NULL,c,d,e,NULL;
```

$$a, b, c, d, e$$

```
> NULL;
```

10.2 Opérations sur les ensembles

Soit les ensembles A={1,2,3}et B:={3,4} :

En mathématiques, contrairement à une liste, l'ordre des éléments ne compte pas et une simplification automatique se produit lorsqu'un même élément apparaît plusieurs fois.

```
> A:={1,2,3}; B:={3,4};
```

$$A := \{1, 2, 3\}$$

$$B := \{3, 4\}$$

```
> c:=A union B; E:= A intersect B; F:= A minus B;
```

$$c := \{1, 2, 3, 4\}$$

$$E := \{3\}$$

$$F := \{1, 2\}$$

$$\{1, 2, 3, z\}$$

10.3 Conversions

La fonction "convert" permet de convertir une liste en ensemble ou l'inverse en utilisant les arguments "set" pour l'ensemble et "list" pour les listes.

Exemple :

```
> x:=[a,b,c];y:=convert(x,set):y;
```

$$x := [a, b, c]$$

$$\{a, b, c\}$$

```
> x:={a,b,c}; y:=convert(x,list):y;
```

$$x := \{a, b, c\}$$

$$[a, b, c]$$

10 .4 Construction automatique de séquences

L'instruction « expr$n » : permet la répétition d'expression expr n fois.

Exemple :

```
> (5!)$5;
```

$$120, 120, 120, 120, 120$$

```
> t:=a$2,y$5;
```

$$t := a, a, y, y, y, y, y$$

Voici un exemple d'utilisation pour répéter la même chose plusieurs fois comme les dérivés :

```
> restart; f := (x^5); diff(f,x); diff(f,x,x);diff(f,x$5);
```

$$f := x^5$$

$$5\,x^4$$

$$20\,x^3$$

$$120$$

$*a* .. *b* : conversion d'intervalle en séquence

```
> [$1..10];
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

***seq* avec plage d'indices**

```
> seq(i^2, i=1..10);
```

1, 4, 9, 16, 25, 36, 49, 64, 81, 100

```
> [seq(ithprime(k),k=1..100)];
```

***seq* avec indices explicites**

```
> x:=[3,5,13]; seq(i^2,i=x);
```

x := [3, 5, 13]

9, 25, 169

```
> seq(i,i="toto");
```

"t", "o", "t", "o"

10 .5 Les Expressions :

L'expression représente le type le plus fréquemment employé dans Maple. Tout ce qui est manipulé dans Maple est considéré comme une expression.

Exemple :
expr1:= x^3;

Comme vu précédemment, Maple traduira immédiatement ces expressions sous forme [opérateur, opérandes].

Il existe diverses fonctions permettant de visualiser cette traduction :

-**whattype**(expression) donne le type général (integer, float, complex, string, expseq, list, set, function, +, *, ^, =,...)

```
> expr1:= x^3+2*x^2-x+5;
```

$$expr1 := x^3 + 2\,x^2 - x + 5$$

```
> whattype(expr1);
```

$$+$$

-**op** (**0**, expression) est plus précise que la précédente

-**op**(expression) donne la liste des opérandes

-**op**(**i**, expression) donne la ième opérande

-**op**(**i..j**,expression) renvois la suite des opérandes du ième au jème

```
> op(0,expr1) ;
> op(expr1);
```

$$+$$

$$x^3,\ 2\,x^2,\ -x,\ 5$$

```
> op (2, expr1);
```

$$2\,x^2$$

```
> op (2.4, expr1);
```

$$2\,x^2,\ -x,\ 5$$

-**nops**(expression) donne le nombre des opérandes

```
> nops(expr1);
```

$$4$$

Une expression peut être modifiée à l'aide des fonctions suivantes :
-**map** (f, expression) où f est une fonction remplace chaque opérande de l'expression par f(opérande)
-**subs** (s1,...,sn,expr) où s1,...,sn sont des équations effectue les substitutions indiquées par les équations dans l'expression

```
> subs(x=z+t,expr1);
```

$$(z+t)^3 + 2\,(z+t)^2 - z - t + 5$$

-subsop(i1=e1,...,in=en,e) où les ik sont des entiers remplace l'opérande de numéro ik par l'expression ek dans l'expression e

```
> subsop(2=x^5,expr1);
```

$$x^3 + x^5 - x + 5$$

Voyons maintenant l'instruction Maple la plus utile : Relancer : cette instruction devrait être la première de tout programme Maple, car elle remet tout dans l'état initial, annule toutes les définitions et permet d'éviter les pièges précédents.

```
> restart:
> P:= x^2+x+1;
```

P := x2 x 1 + +

Ceci est une expression pas une fonction ; elle n'est pas évaluée car la variable x est libre.

```
> x:=1: P; 3> x:=2: P; 7
 > Q := x^4+x^2+1; := Q21
```

Q est évalué complètement car x n'est plus libre.

> x := 'x'; # on libère x := xx
> P ; Q ; ++x2x1 21 On constate que P a 'retrouvé la mémoire', mais pas Q.

11 Equations et systèmes d'équations

Pour résoudre une équation en utilisant la commande solve de Maple, une équation est de la forme "expression = 0".

Exemple :

```
> equ:=x^2+x-2=0;
```

$$equ := x^2 + x - 2 = 0$$

```
> solve(equ,x);
```

1, -2

En remarque que Maple recherche l'ensemble des solutions dans R ou dans C et les place dans une séquence. On peut également affecter les solutions à une liste ou à un ensemble.

Solutions : = [solve (equ, var)], solutions est le nom de la liste des solutions où :

Solutions : = {solve (equ, var)} solutions est le nom de l'ensemble des solutions.

Noté que les solutions multiples ne sont figurées qu'une seule fois.

Exemple :

```
> equ:= cos(x^2)+sin(2*x^2);
```

$$equ := \cos(x^2) + \sin(2\,x^2)$$

```
> sol_list:=[solve(equ,x)];
```

$$sol\_list := \left[\frac{1}{2} I\sqrt{2}\sqrt{\pi}, \frac{-1}{2} I\sqrt{2}\sqrt{\pi}, \frac{\sqrt{2}\sqrt{\pi}}{2}, -\frac{\sqrt{2}\sqrt{\pi}}{2}, \frac{1}{6} I\sqrt{6}\sqrt{\pi}, \frac{-1}{6} I\sqrt{6}\sqrt{\pi}, \frac{1}{6} I\sqrt{30}\sqrt{\pi}, \frac{-1}{6} I\sqrt{30}\sqrt{\pi}\right]$$

```
> sol_set:={solve(equ,x)};
```

$$sol\_set := \{\frac{\sqrt{2}\sqrt{\pi}}{2}, -\frac{\sqrt{2}\sqrt{\pi}}{2}, \frac{-1}{6} I\sqrt{30}\sqrt{\pi}, \frac{1}{6} I\sqrt{30}\sqrt{\pi}, \frac{-1}{6} I\sqrt{6}\sqrt{\pi}, \frac{-1}{2} I\sqrt{2}\sqrt{\pi}, \frac{1}{2} I\sqrt{2}\sqrt{\pi}, \frac{1}{6} I\sqrt{6}\sqrt{\pi}\}$$

Dans le cas d'un système, toutes les équations et toutes les variables doivent être placées dans un ensemble.

```
> eq1:=2*x+y=0;eq2:=x+y-z=0;eq3:= x+y+z=-2;
```

$$eq1 := 2x + y = 0$$

$$eq2 := x + y - z = 0$$

$$eq3 := x + y + z = -2$$

```
> solve({eq1,eq2,eq3},{x,y,z});
```

$$\{x = 1, y = -2, z = -1\}$$

12 Les fonctions

1) les méthodes de définition :

-par la flèche : **f : =var->expr.**

S'il y a plusieurs variables : **f : = (var1, var2,...) ->expr ;**

Exemple

```
> f:=x->ln(x^2+2)-sqrt((x-1));
```

$$f := x \to \ln(x^2 + 2) - \sqrt{x - 1}$$

```
> g:=(x,y)->x^2+y^2;
```

$$g := (x, y) \to x^2 + y^2$$

2) domaines de définition, de continuité :

Les deux fonctions suivantes :

-singular (fonct, var) permet de déterminer les singularités d'une fonction

-iscont (fonct, x=a.. b,'closed') permet d'étudier la continuité de la fonction sur l'intervalle [a,b]- fermé avec l'option closed- ou]a,b[-sans l'option closed.

Exemple :

```
> singular(f(x),x);
```

$$\{x = \sqrt{2}\ I\}, \{x = -I\sqrt{2}\}, \{x = \infty\}, \{x = -\infty\}$$

```
> iscont(f(x),x= -infinty..0);
```

true

3) image d'intervalle, extréma:

Les fonctions evalr, minimize et extrema sont à utiliser.

-evalr(fonction([a,b])) détermine l'image de l'intervalle [a,b]par fonction

```
> evalr(f(INTERVAL(3..10)));
```

$$\text{INTERVAL}(-\sqrt{9} + \ln(11) .. -\sqrt{2} + \ln(102))$$

-minimize (fonction, {vars}, intervalle) retourne le minimum de fonction par rapport à la variable vars dans intervalle.

-maximize (fonction, {vars}, intervalle) retourne le maximum

-extrema (fonction, {contraintes}, var,'s') donne les extrema de fonction par rapport à var en tenant compte des contraintes - s'il n'y a pas de contraintes, écrire {} -. Les valeurs correspondantes de var sont stockées dans s.

13 Derivees &Integrales:

1)Dérivée:

-diff(f,x) dérivée la fonction a par rapport la variables x.

Exemple :

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

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

Lorsque l'instruction (diff) est écrite avec une majuscule, il s'agit de la forme inerte (Diff), c'est à dire que l'expression est affichée mais non exécutée.

```
> Diff(f(x),x)=diff(f(x),x);
```

$$\frac{d}{dx}(\ln(x^2+2) - \sqrt{x-1}) = \frac{2x}{x^2+2} - \frac{1}{2\sqrt{x-1}}$$

Il existe d'autre forme de dérivée come :
-diff(a,x1,x2...xn) dérivée de l'expression ou fonction a par rapport aux variables x1,x2...xn successivement

-diff(a,x$n), dérivée d'ordre n de a par rapport à x.$ est appelé opérateur séquentiel

2) intégrales :

-int(f,x) retourne la primitive de f. Maple n'affiche pas la constante d'intégration. Int en est la forme inerte.

```
> Int(f(x),x)=int(f(x),x);
```

$$\int \ln(x^2+2)-\sqrt{x-1}\,dx = x\ln(x^2+2)-2x+2\sqrt{2}\arctan\left(\frac{x\sqrt{2}}{2}\right)-\frac{2(x-1)^{(3/2)}}{3}$$

-int(f,x=a..b,options) intégrale définie de f entre a et b

```
> Int(f(x),x=-1..1)=int(f(x),x=-1..1);
```

$$\int_{-1}^{1} \ln(x^2+2)-\sqrt{x-1}\,dx = 2\ln(3)-4+4\sqrt{2}\arctan\left(\frac{\sqrt{2}}{2}\right)-\frac{4}{3}I\sqrt{2}$$

-**changevar**(chang,int,u) faire **with**(Student) avant. Effectue le changement de variable dans l'intégrale int, chang est de la forme f(x)=g(u) où x est l'ancienne variable et u la nouvelle.

-**intparts**(int,u) faire **with**(student) avant. Fais l'intégration par parties, étant le facteur à dériver dans int.

14 les polynômes :

14.1 Manipulation des polynômes :

Maple enregistre des expressions connues sous le nom de polynômes. Bien qu'on puisse utiliser les fonctions op(p) ou op (i, p) pour isoler des opérandes, il offre des fonctions spécifiques aux polynômes pour simplifier leur manipulation. Étant capable de travailler avec des polynômes à plusieurs variables, Maple requiert la spécification de la variable sur laquelle une fonction sera appliquée.

12.2 Manipulation de termes :

-coeff(p,x,n) retourne le coefficient de x^n si le polynôme est ordonné.

-coeffs(p,x,'t') génère une séquence contenant tous les coefficients. 't' est facultatif. Maple lui associe les x^i dans le même ordre que les coefficients de la séquence. Cette fonction s'avère utile lorsque certains coefficients sont nuls.

-lcoeff(p,x,'t') donne le coefficient de plus haut degré en x

-tcoeff(p,x,’t’) renvoie le coefficient de degré le plus bas. L’utilisation de ’t’ est facultative dans ces deux fonctions.

-degree(p,x) renvoie le degré en x du polynôme

-ldegree(p,x) renvoie le plus bas degré du polynôme

12 .3 Regroupement de termes:

-collect(a,x,g,f) regroupe les coefficients ayant le même degré en x. Les paramètres g et f sont facultatifs. Pour les polynômes à plusieurs variables l’option ‘distributed’ doit être notée.

-l’usage de la fonction normal(p) présente le polynôme p sous la forme classique

-Avec la fonction sort (p,options) vous pouvez ordonner un polynôme en respectant les options spécifiées.

-expand(p) c’est la fonction qui distribue les produits sur les sommes pour developper le polynôme p.

-convert(p,horner) pour factoriser p sous la forme d’un polynôme de Horner

-compoly(r,x) une fonction qui retourne p(x) et x=q(x) tels que subs(x=q(x),p(x))=r(x).dont, Il

Est nécessaire de lire la bibliothèque « compoly » via la commande readlib(compoly);

14.4 opérations arithmétiques sur les polynômes:

-+,-,*,^ addition, soustraction, multiplication et puissance

-quo(a,b,x,’r’) pour calculer le quotient de a par b, avec un reste stocké dans r

-rem(a,b,x,’q’) qui calcule le reste de la division euclidienne de a par b, le quotient est stocké dans q

-gcd(a,b,’cofa’,’cofb’) calcule le pgcd de a et b, cofa et cofb sont optionnels:ce sont les cofacteurs cofa=a/gcd(a,b)

-lcm(a,b) calcule le ppcm de a et b.

-content(a,x,’pp’) calcule le pgcd des coefficients de a.pp vaut a/pgcd(coeff)

-primpart(p,x) donne le résultat de p/content(p,x) comme le pp précédent

-divide(a,b,'q') renvoie TRUE si la division de a par b est exacte en stockant le quotient dans q, FALSE si la division n'est pas exacte, q n'est alors pas affecté.

-subs (x=expression,p) elle remplace la variable x par l'expression donnée dans le polynôme p. cette fonction est utile pour évaluer p en une valeur spécifique de x.

-norm(p,n) calcule la norme d'ordre n du polynôme. Cela correspond à la racine nième de la somme des coefficients de p, chacun élevé à la puissance n.

-powmod(a,n,b,x) retourne a^n mod b

-randpoly génère un polynôme de manière aléatoire

-interp(x,y,v) pour obtenir le polynôme d'interpolation de Lagrange à partir d'un ensemble de points dont les coordonnées sont dans x et y, présenté sous forme de liste.

15 Le graphisme sous Maple

Il existe trois types de graphisme 2D, 3D et

15.1 GRAPHISME 2-D

Il existe plusieurs commandes pour tracer des courbes en deux dimensions, chacune étant adaptée à une forme spécifique de fonction : explicite, implicite, paramétrique ou polaire. Le choix de la commande à utiliser dépend de la nature de la fonction que vous souhaitez représenter.

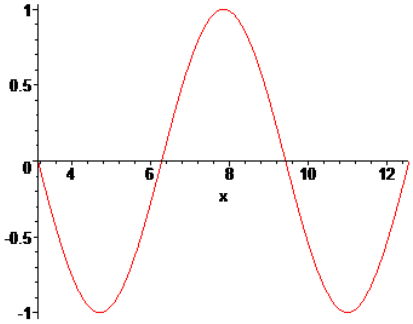
Forme explicite

Soit à dessiner la fonction f = f(x) sur l'intervalle [a , b].

> plot(f(x), x = a..b);

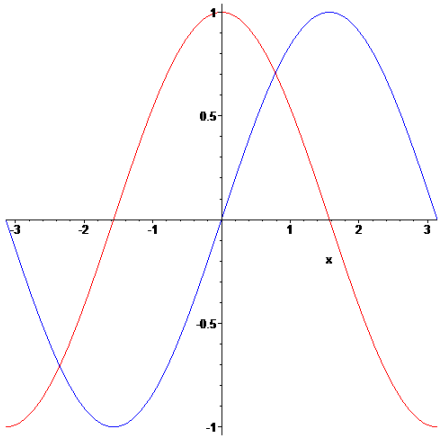
Ex : Dessiner la fonction f(x) = sin(x) sur [π , 4 π]

```
> plot( sin( x ), x = Pi..4*Pi);
```



Ex : Dessiner sur un même graphique les fonctions f (x) et g(x) ci-dessous, dont f(x) sera en bleu et g(x) en rouge, sur l'intervalle [-π, π] f(x) = sin(x) g(x) = cos(x)

```
> plot([sin(x), cos(x) ], x = -Pi..Pi, color = [blue, red] );
```



Rem: [sin(x),cos(x)] étant une liste de même que [blue, red] ainsi l'ordre des fonctions à dessiner suivra l'ordre de la liste des couleurs.

Forme implicite

Soit représenter la fonction implicite f (x, y) = 0. Le module plots doit être introduit pour cela. Comme tel :

>with(plots); [>implicit (f (x, y), x = a ..b, y = c..d);

Ex : Dessiner la courbe dont l'équation cartésienne est e cos(x y) xy = − pour 0 ≤ x ≤ π et 2 ≤ y ≤ 4 Noter que y = y (x).

> with(plots);

> implicitplot(exp(x*y) - cos(x-y), x = 0..Pi , y = 2..4);

Forme paramétrique

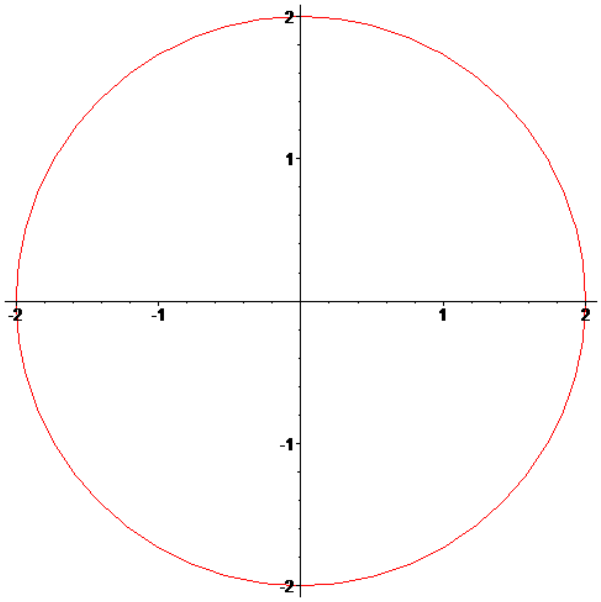
Il faut tracer une fonction sur l'espace [a, b] et utiliser les équations paramétriques suivantes : x = f (t) et y = g (t).

La syntaxe de la commande est la suivante :

> plot([f (t), g (t), t= a..b], scaling = constrained);

Ex: Soit à tracer un cercle de rayon R = 2 , sur l'intervalle [0..2 π] , dont les équations paramétriques sont: x(t) = 2 *cos(t) y(t) = 2* sin(t)

```
> plot( [ 2*cos(t), 2*sin(t), t = 0..2*Pi], scaling =constrained );
```



Forme polaire

Soit à tracer la cardioïde dont l'équation polaire est r(ß) = 1+cos(ß), sur [0 , 2 π], dans le système de coordonnées cartésiennes.

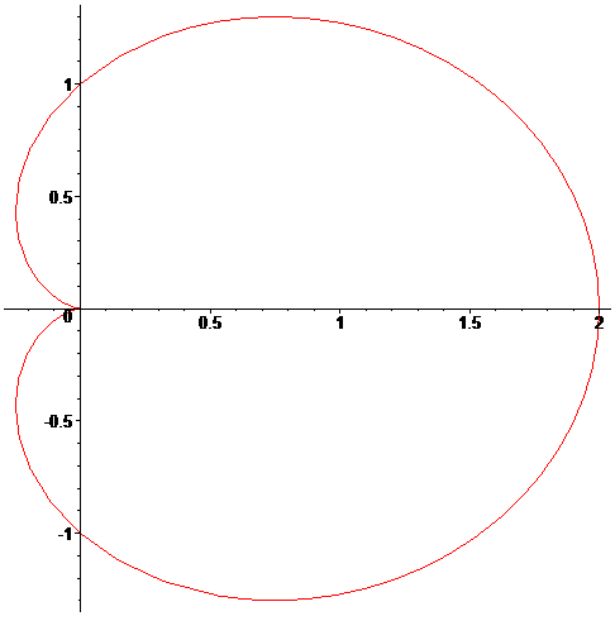
>with(plots);

Le module plots est présenté ici.

```
> r := 1+cos(beta);
```

$$r := 1 + \cos(\beta)$$

```
> polarplot( r, beta = 0..2*Pi );
```



>

15.2 GRAPHISME 3-D

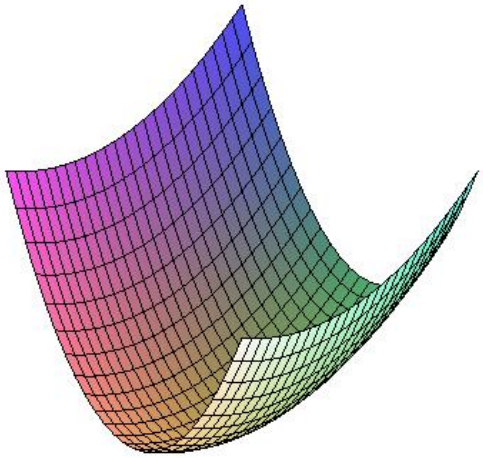
Forme explicite

Il faut créer une surface avec l'équation z = f (x, y). La syntaxe de la commande est la suivante :

> plot3d(f (x , y) , x = a..b, y = c..d);

Ex : Soit à dessiner la surface z = x2 + y2 pour $-1 \leq x \leq 1$ et $-2 \leq y \leq 2$

```
> plot3d (x^2+y^2, x = -1..1, y = -2..2 );
```



>

Forme implicite

Il faut représenter la surface S une fois que l'équation de celle-ci est F(x,y,z)=0. Les commandes sont les suivantes :

>with(plots);

> implicitplot3d (F(x ,y , z) , x = a..b, y = c..d , z = e..f);

Forme paramétrique Soit à dessiner une courbe dans l'espace dont on connaît les équations paramétriques suivantes : x = f(t) y = g(t) $a \leq t \leq b$ z = h(t).

Les commandes sont :

> with(plots);

> spacecurve([f(t), g(t), h(t)], t = a..b);

Ex: Soit à tracer dans l'espace l'hélice circulaire dont les équations paramétriques sont: x = cos(t) y = sin(t) $0 \leq t \leq 2\pi$ z = t Les commandes sont:

> with(plots);

```
> spacecurve( [ cos ( t ), sin ( t ), t ], t = 0..2*Pi);
```

>

Si l'on veut représenter une surface en utilisant les équations paramétriques suivantes :

x = f (t , s) y = g (t , s) $a \leq t \leq b$ et $c \leq s \leq d$ z = h (t , s) La commande est

> plot3d([f (t , s) , g (t , s), h (t , s)] , t = a..b , s = c..d);

15.3 Graphisme en coordonnées cylindriques et sphériques

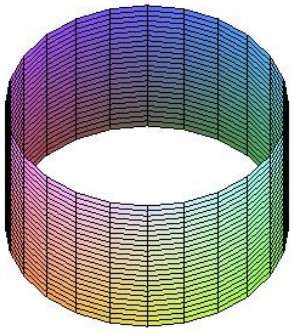
Si l'équation de la surface à tracer est exprimée dans des coordonnées cylindriques ou sphériques, vous pouvez également utiliser la commande plot3d avec l'option coords. Cette option peut être définie comme 'cylindrical' si vous utilisez des coordonnées cylindriques, ou comme 'spherical' si vous utilisez des coordonnées sphériques.

En coordonnées cylindriques La distance r, entre un point d'une surface et l'axe z est exprimé par : r = r (θ, z) avec $\theta 1 \leq \theta \leq \theta 2$ et $1\ 2\ z \leq z \leq z$ alors la commande est :

> plot3d (r (θ, z), theta = 0..2*Pi, z =z1..z2, coords = cylindrical, scaling = constrained);

Ex : Dessiner un cylindre de rayon r = 5 et de hauteur 6 en coordonnées cylindriques

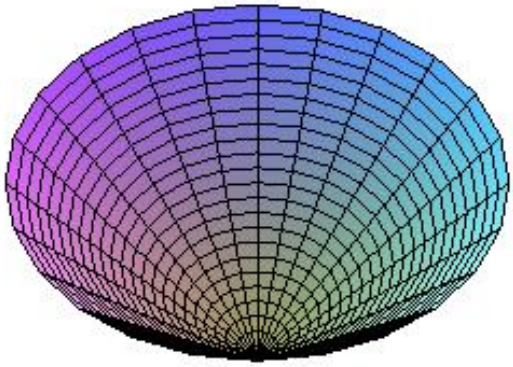
```
> plot3d ( 5, theta = 0..2*Pi, z = 0..6, coords = cylindrical ,
scaling = constrained );
```



>

Ex: Dessiner un cône d'équation 2 2 2 z = x + y de hauteur 7 en coordonnées cylindriques.

```
> plot3d ( z, theta = 0..2*Pi, z = 0..7, coords = cylindrical, scaling
= constrained );
```



>

En coordonnées sphériques La distance entre un point de la surface et l'origine est exprimée par : ρ = ρ (θ, φ) avec θ 1 ≤ θ ≤ θ 2 et φ1 ≤ φ ≤ φ 2 alors la commande est/

> plot3d (ρ (θ, φ), theta = θ1.. θ2, phi = phi1..phi2, coords = spherical , scaling = constrained);

Ex: Dessiner une sphère de rayon 5 centrée à l'origine, en coordonnées sphériques.

> plot3d (sqrt(25-z^2), theta = 0..2*Pi, phi = 0..Pi, coords = cylindrical , scaling = constrained);

16 Programmation

Remarque importante : afin d'entrer plusieurs lignes d'instructions Maple successives sans les exécuter à la fin de chaque ligne, utilisez la combinaison de touches Shift+Entrée à la fin de chaque ligne. Ensuite, pour valider l'ensemble des lignes rédigées, appuyez sur Entrée en plaçant le curseur, par exemple, sur la dernière ligne que vous avez écrite.

16.1 LES STRUCTURES DE test : if .. then

Trois possibilités sont disponibles

1)
if condition1 ***then***
instruction 1
….
end if;
	Nombre paire
	`> a:=14:`
	`> if irem(a,2)=0 then`
	`print(a,` est Un nombre paire`)`
	`end if;`
	14, *est Un nombre pair*

2)
if condition ***then***
instruction 1
….
instruction n
else
instructions
end if;
	Nombre paire ou impair
	`> a:=13:`
	`> if irem(a,2)=0 then`
	`print(a,` est Un nombre paire`)`
	`else`
	`print(a,` est Un nombre impaire`)`
	`end if;`
	13, *est Un nombre impair*

3)
if condition ***then***
instruction 1
elif condition2 ***then***
instruction2
elif condition3 ***then***
instruction3 …
	résolution de l'équation du premier degré
	`> a:=0:b:=0:`
	`> if a<>0 then print(`Une solution : x `=-b/a)`
	`elif b=0 then print(`Tout x est solution`)`
	`else print(`Pas de solution`)`
	`end if;`
	Tout x est solution

else instruction ***end if;***	

16.2 Les boucles

boucle while .. do:
 	
while condition ***do***
 	
Séquence d'instructions
 	
end do;
 	
# Tant que la condition est vraie, exécute une suite d'instructions. 	
	Exemple: calcul de la somme des 100 premiers entiers naturels
	
	`> somme:=0:k:=1:`
	`while k<=100 do`
	`somme:=somme+k:`
	`k:=k+1:`
	`end do:`
	`print(`somme =`,somme);`
	
	$somme = 5050$

Structure de contrôle for .. to:
 	
for variable ***from*** initiale ***to*** finale ***by*** pas ***do***
 	
…
 	
end do;
 	
Exécute une boucle pour une variable allant d'une valeur initiale à une valeur finale, avec un pas donné.
 	
Or
 	
for variable ***in*** expression ***do*** .. ***end do;***
 	
Exemple afficher les nombres premiers de la liste [17, 19 , 31,39,47]
 	
`> for k in [17,19,30,39,47] do`
`if isprime(k) then`
`print(k,`est premier`)`
	Reprenant l'exemple précédent
	
	`> somme:=0:`
	`for k from 1 to 100 do`
	`somme:=somme+k`
	`od:`
	`print(`somme =`,somme);`
	
	$somme = 5050$
	
	la somme des entiers impairs inférieurs à 100
	
	`> somme:=0:`
	`for k from 1 by 2 to 100 do`
	`somme:=somme+k`
	`od:`
	`print(`somme =`,somme);`
	
	$somme = 2500$

fi end do; 17, *est premier* 19, *est premier* 47, *est premier*	

16. 3 LES PROCÉDURES:

Les sous-programmes d'une procédure sont définis par le mot-clé "proc" dans MAPLE et peuvent être assignés à un nom de variable. Voici la syntaxe d'une procédure Maple :

Nom_procedure := ***proc*** (paramètres_formels)

global variables_globales; (déclaration des variable ***global.*** Cette ligne est optionnelle)

local variables_locales; (déclaration des variables ***local***. Cette ligne est optionnelle)

description chaîne_de_description; (la ligne ***description*** est optionnelle)

option nom_option; (la ligne ***option*** est optionnelle)

. . . instructions . . . (corps de la procédure)

end proc;

Au sein d'une procédure, seules les commandes 'proc' et 'end proc' sont obligatoires. Toutes les autres instructions sont flexibles et dépendent des besoins. Les arguments ou inputs de la procédure sont désignés comme des paramètres. Les éléments constitutifs d'une procédure sont explicités dans les sections à venir.

16.4 Les types des arguments dans une procédure

Souvent, les arguments d'une procédure sont associés à des types spécifiques. Pour imposer une contrainte de type sur un argument xi, on peut ajouter cette contrainte directement après xi dans la première ligne de la procédure, en utilisant la syntaxe xi :: typei. Par exemple, pour contraindre l'argument d'entrée n à être un nombre dans la procédure moitie, on écrirait :

```
>moitie := proc(n :: numeric) trunc(n/2) ;
end proc ;
                moitie := proc(n :: numeric)trunk(1/2*n) end proc
>moitie(a);
          Error, invalid input : moitie expects its 1st argument, n, to be
                          of type numeric, but received a
```

16.5 Variables locales et variables globales

L'instruction locale de la procédure f permet de définir une liste de variables dont la portée est restreinte à cette procédure. Si l'on modifie la procédure moitié pour attribuer à la variable b la valeur n/2, cela s'appliquera uniquement à l'intérieur de cette procédure.

```
b :=    100 ;
moitie := proc(n) local b ;
b := n/2 ; trunc(b) ; end proc ;
moitie := proc(n)local b ;b :=1/2*n ; trunc(b) end proc
>moitie(5) ;
  >b ;
                              2

                             100
```

On observe que la variable 'b' possède une valeur locale égale à la moitié de n/2 à l'intérieur de la procédure et une valeur globale de 100.

La commande 'globale' fournit la liste des variables ayant la même valeur à la fois à l'intérieur et à l'extérieur d'une procédure.

```
c := 70 ;
b := 100 ;
moitie := proc(n)
local b ; global c ;
b := trunc(n/2) ; c := 374 ; print(b) ; print(c) ;
end proc :
moitie (5) ;


b ;

c ;

                              2
                             374
```

100

374

Exemples :

```
# procédure qui donne le factoriel de n
# procédure qui calcule la somme des entiers de 1 à n
somme := proc(n)    # ici pas de point virgule local s,i;        #
variables locales s := 0;
for i from 1 to n do s := s + i;
             od:
             s;
end;
```

somme := **proc** (*n*) **local** *s*, *i*; *s* := 0; **for** *i* **to** *n* **do** *s* := *s* + *i* **end do** ; *s* **end proc**

```
> somme(10);
```

55

```
> somme(100);
```

5050

```
> # procédure qui donne le factoriel de n restart;
> fact := proc(n)
local p,i; p:=1;
for i from 1 to n do p := p*i;
od;
p;
end;
```

facto := **proc** (*n*) **local** *p*, *i*; *p* := 1; **for** *i* **to** *n* **do** *p* := *p*×*i* **end do** ; *p* **end proc**

```
> fact(5);facto(10);
```

120

3628800

```
> t:=time(): facto(300): time()-t;
                                .001

> t:=time(): facto(30000): time()-t;
                               17.356
```

La fonction en Maple qui donne le nombre de bits nécessaire pour coder un entier n en binaireest :

```
> nbits := proc(n)
     local nb, i;
     i:=iquo(n,2)
    nb:=1;
    while(i<>0) do
    i:=iquo(i,2); /* le quotient de i par 2 */ nb:=nb + 1;
    od;
    nb;
    end;
```

- La fonction Maple iquo(n,k) renvoie le résultat de la division entière (quotient) de n par k.
- exemple : iquo(7,2) donne 3.
- L'expression n mod k donne le reste de la division de n par k. exemple : 7 mod 2 donne 1.

16.6 Valeur de retour d'une procédure

La valeur résultante d'une procédure f est déterminée par la dernière évaluation réalisée à l'intérieur de cette procédure. Les instructions RETURN et ERROR offrent la possibilité de changer ce comportement. L'utilisation de la commande ERROR(message) arrête l'exécution de la procédure et affiche le message spécifié, empêchant toute valeur de retour de la procédure.

Exemple :

```
moitie := proc(n)
if not(type(n,numeric)) then
ERROR('l'argument doit être de type numérique') else
trunc(n/2) ;
end if ; end proc ;
moitie(x) ;
    ERROR, (in moitie) l'argument doit être de type
    numérique
```

La commande RETURN(x) placée à l'intérieur d'une procédure interrompt son exécution et attribue à x la valeur de retour de la procédure.

Exemple : Appartenance à une liste – la procédure suivante vérifie si une expression x appartient à une liste L.

```
element := proc(x :: anything, L :: list) local i;
for i to nops(L) do
if L[i]=x then RETURN(vrai) end if; end do;
faux;  end proc;
>element(a,[a,b,e,a]);
                        VRAI
>element(c,[a,b,e,a]);
                         FAUX
```

16.7 Procédures récursives

Une procédure récursive s'appelle :

Exemple : calcul du factoriel d'un nombre entier

```
facteur := proc(n)
 local f;
if n<>1 then
f:=n*facteur(n-1); else return(n);
```

```
end if;
end proc;
```

facteur := proc(*n*) local *f;* if *n* <> 1 then *f* := *n***facteur*(*n* - 1) else return *n* end if*;* end proc*;*

```
facteur(3);
```

6

16.8 Sauvegarde de procédures et de calculs

Si vous souhaitez préserver des procédures élaborées pendant une session de travail ou conserver le résultat d'un calcul prolongé, Maple offre la possibilité de sauvegarder ce travail au format interne de Maple. Pour ce faire, vous pouvez utiliser la commande ‘save’ qui copie le texte dans un fichier portant l'extension .m. La syntaxe de cette commande est :

```
Save listenoms,fichier.m ;
```

Exemple :

Nous souhaitons enregistrer la procédure "facteur" de l'exemple précédent ainsi que le résultat de cette procédure dans un fichier nommé "fichier_facteur(5)". Pour ce faire, nous effectuons les actions suivantes :

```
> n := facteur(5) :
      save facteur,n,"fichierfact.m ";
```

Lorsqu'on exécute une commande sur Maple qui ne produit pas de réponse directe, comme dans le cas de la création d'un fichier nommé "fichierfact.m", Maple n'affiche aucune sortie immédiate. Cependant, ce fichier est créé et son contenu peut être récupéré ultérieurement lors d'une session en utilisant la commande read.

```
restar;
n ;
read " fichierfact.m";n
```

120

17 CALCUL ALGEBRIQUE (vecteur et matrice)

Le traitement des vecteurs et des matrices sous Maple se fait par deux approches différentes. Premièrement, ils peuvent être considérés comme des tableaux à une ou deux dimensions, ce qui implique l'application des opérations standard pour les tableaux. Deuxièmement, des fonctions spécifiques à Maple ont été développées pour ces structures et sont regroupées dans la bibliothèque "linalg". Pour utiliser ces fonctions, l'utilisateur doit préalablement indiquer l'utilisation de cette bibliothèque. Sans cette indication, les fonctions correspondantes ne seront pas accessibles.

17.1 Manipulation des vecteurs

Comme mentionné précédemment, Maple gère les vecteurs de deux manières distinctes :

- Un tableau monodimensionnel qui obéit aux opérateurs usuels manipulant des tableaux,

- Une variable de type vecteur « vector » après le chargement de la bibliothèque spéciale « linalg » à titre d'exemple, un vecteur vu sous forme d'un tableau est déclaré comme suit : Les composants d'un tel vecteur sont indicés de 1,2 … Avec le package « linalg », un vecteur est déclaré selon quatre formes :

1. vector([x1, ..., xn])
2. vector(n, [x1, ..., xn])
3. vector(n)
4. vector(n, f)

Où les paramètres sont définis comme suit :

- x1, ..., xn sont les éléments du vecteur,
- n la dimension d'un vecteur,
- f facultatif : pour initialiser les éléments du vecteur.

Exemple :

```
> V:=array([3,7,4,6]);
```

$$V := [3, 7, 4, 6]$$

```
> V[2];
```

$$7$$

```
> V:=vector(4,[3,7,4,6]);
```

$$V := [3, 7, 4, 6]$$

```
> w:=vector(4);
```

$$w := \text{array}(1 .. 4, [\ ])$$

```
> s:=vector(4,5);
```

$$s := [5, 5, 5, 5]$$

```
> t:=vector(4,rand(15));
```

$$t := [1, 8, 5, 8]$$

Le paramètre f a été obtenu à l'aide de la fonction rand(20), générant un nombre aléatoire dans la plage de 0 à 20. Cette méthode permet de déclarer un vecteur de manière aléatoire, bien qu'il soit également envisageable de le définir dès le début. Parmi les fonctions dédiées à la manipulation des vecteurs, il existe un large éventail d'instructions disponibles. Il est recommandé de charger le package "linalg" pour accéder à ces fonctionnalités.

A titre d'exemple, les instructions suivantes sont souvent utilisées.

- angle (V,W) : retourne l'angle en radians entre les vecteurs V et W,
- dotprod (V,W) : calcule le produit scalaire de V par W,
- crossprod (u,v) : calcule le produit vectoriel de V par W,
- norm(v) : calcule la norme du vecteur V,
- normalize (V) : normalise le vecteur V,
- randvector (n) : génère un vecteur de dimension n à éléments aléatoires.
- vectdim (V) : renvoie la dimension du vecteur V.

17.2 Manipulation des matrices

Les matrices sont principalement considérées comme des tableaux à deux dimensions, permettant ainsi l'utilisation des opérations de tableau standard. Dans cette configuration, les éléments des matrices peuvent être des expressions Maple variées, y compris des chaînes de caractères, et sont accessibles via les indices "i" et "j" pour parcourir les lignes et colonnes. Le package "linalg" fournit des opérations spécifiques aux matrices, donc travailler avec ce package est recommandé pour manipuler efficacement les matrices. Il regroupe les diverses fonctions nécessaires pour les calculs et manipulations des matrices. Les indices d'une matrice doivent être des entiers strictement positifs, et les éléments doivent être des expressions mathématiques valides, ce qui les différencie des tableaux classiques.

Déclaration générale d'une matrice :

Les matrices sont déclarées par l'instruction matrix () selon la syntaxe suivante :

- matrix (Lv)
- matrix (m, n)
- matrix (m, n, Lv)
- matrix (m, n, f)

Les paramètres des instructions mentionnées ci-dessus ne sont pas tous facultatifs, mais Lv, m et n sont parmi les plus cruciaux. Lv représente la liste des vecteurs constituant les colonnes de la matrice, tandis que m et n déterminent sa dimension. Dans cet exemple, les éléments de la matrice ne sont pas encore définis. En définissant ces trois paramètres, on établit la structure de la matrice sans spécifier ses éléments. Cependant, ces éléments peuvent être initialisés avec des valeurs spécifiques, par exemple, des nombres aléatoires entre 0 et 30. Il n'est pas possible d'initialiser la matrice en utilisant simplement l'instruction matrice(m,n) sans spécifier les valeurs. Ainsi, la matrice reste vide, sans aucune valeur définie.

Une matrice vide peut être utile pour la remplir ultérieurement via une programmation spécifique ou en lisant des données à partir d'autres fichiers. Cependant, il est également avantageux de créer une matrice initialisée dès le départ avec des zéros ou des nombres aléatoires. En outre, une autre approche consiste à sélectionner une liste de vecteurs comme suit :

M :=matrix(n,m,[[v11,…v1m],[v21,…,v2m],….. [vn1,….vnm]]) ;

Lorsque vous déclarez une matrice dans Maple, elle n'est pas affichée sous sa forme matricielle usuelle par défaut. Pour visualiser la matrice dans le format habituel, vous devez utiliser la fonction d'évaluation des matrices, evalm().

17.3 Opérations courantes sur les matrices

À la différence des variables scalaires, les matrices doivent être déclarées avant leur utilisation. Les opérateurs standards agissant sur les matrices sont donnés comme suit :

- + : la somme
- - : la soustraction,
- &* : le produit,
- ^ ou ** : la puissance.

Exemple : Soit les deux matrices A et B suivantes :

```
> A:=matrix([[1,3,6],[0,8,6],[10,3,7]]);
```

$$A := \begin{bmatrix} 1 & 3 & 6 \\ 0 & 8 & 6 \\ 10 & 3 & 7 \end{bmatrix}$$

```
> B:=matrix([[0,5,8],[6,8,1],[4,3,5]]);
```

$$B := \begin{bmatrix} 0 & 5 & 8 \\ 6 & 8 & 1 \\ 4 & 3 & 5 \end{bmatrix}$$

```
c
```

```
> A;
```

$$A$$

```
> A+B;
```

$$A + B$$

```
> evalm(A+B);
```

$$\begin{bmatrix} 1 & 8 & 14 \\ 6 & 16 & 7 \\ 14 & 6 & 12 \end{bmatrix}$$

Pour obtenir le produit de deux matrices, utilisez l'opérateur "&" plutôt que l'opérateur standard "*". Cela simplifie grandement la procédure.

```
> E:=A&*B;
```

$$E := A \&* B$$

```
> evalm(E);
```

$$\begin{bmatrix} 42 & 47 & 41 \\ 72 & 82 & 38 \\ 46 & 95 & 118 \end{bmatrix}$$

L'application d'une fonction à une matrice, ainsi que l'élévation en puissance d'une fonction, peut être réalisée de manière individuelle sur ses éléments. L'exemple ci-dessous illustre le calcul d'une puissance et l'application de la fonction exponentielle sur une matrice A définie au début de cet exemple.

```
> evalm(A^3);
```

$$\begin{bmatrix} 721 & 741 & 1098 \\ 960 & 1106 & 1482 \\ 1350 & 1221 & 1819 \end{bmatrix}$$

```
> evalm(exp(A));
```

$$\begin{bmatrix} \mathbf{e} & \mathbf{e}^3 & \mathbf{e}^6 \\ 1 & \mathbf{e}^8 & \mathbf{e}^6 \\ \mathbf{e}^{10} & \mathbf{e}^3 & \mathbf{e}^7 \end{bmatrix}$$

Maple offre un accès direct aux opérateurs fondamentaux, mais la manipulation des matrices va bien au-delà de cette approche de base. Pour cela, Maple met à disposition sa bibliothèque spéciale, le package "linalg".

17.4 Les fonctions du package linalg

Le package "Linalg" offre une variété de fonctions dédiées à la manipulation et à l'exploitation des vecteurs et des matrices. Pour bénéficier de ces fonctionnalités, il est nécessaire de charger le package "Linalg" en suivant les instructions suivantes :

```
> with(linalg);
Warning, the protected names norm and trace have been redefined and
unprotected
```

[*BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian, addcol, addrow, adj, adjoint, angle, augment, backsub, band, basis, bezout, blockmatrix, charmat, charpoly, cholesky, col, coldim, colspace, colspan, companion, concat, cond, copyinto, crossprod, curl, definite, delcols, delrows, det, diag, diverge, dotprod, eigenvals, eigenvalues, eigenvectors, eigenvects, entermatrix, equal, exponential, extend, ffgausselim, fibonacci, forwardsub, frobenius, gausselim, gaussjord, geneqns, genmatrix, grad, hadamard, hermite, hessian, hilbert, htranspose, ihermite, indexfunc, innerprod, intbasis, inverse, ismith, issimilar, iszero, jacobian, jordan, kernel, laplacian, leastsqrs, linsolve, matadd, matrix, minor, minpoly, mulcol, mulrow, multiply, norm, normalize, nullspace, orthog, permanent, pivot, potential, randmatrix, randvector, rank, ratform, row, rowdim, rowspace, rowspan, rref, scalarmul, singularvals, smith, stackmatrix, submatrix, subvector, sumbasis, swapcol, swaprow, sylvester, toeplitz, trace, transpose, vandermonde, vecpotent, vectdim, vector, wronskian*]

Lorsque la bibliothèque est chargée, la liste des différentes fonctions prises en charge est affichée, accompagnée de deux avertissements initiaux concernant la redéfinition des fonctions norm et trace. Les fonctionnalités offertes par le package "linalg" sont variées. Pour débuter, il est recommandé de se concentrer sur les fonctions dont le concept est plus simple.

17.4.1 Matrices particulières:

Les matrices spécifiques servent de raccourcis vers des structures matricielles très pratiques et couramment utilisées. Elles évitent souvent l'utilisation de lignes de commande complexes, simplifiant ainsi la résolution des problèmes. Les instructions suivantes détaillent ces cas spécifiques.

- diag(B1,B2,...Bn) génère une matrice en plaçant les matrices B1,B2...Bn sur la diagonale. La matrice obtenue est diagonale par blocs. Dans l'exemple à droite, la matrice A et B sont utilisée pour former la matrice diagonale C.

- genmatrix(eqns,vars) : engendre une matrice à partir des coefficients des variables vars dans le système d'équations linéaires eqns.

- jacobian(V,u) : calcule la matrice jacobéenne du vecteur V par rapport aux coordonnées stockées dans le vecteur u.

- randmatrix(m,n) : conçois une matrice aléatoire de dimension mxn.

```
> A:=matrix([[3,2],[4,6]]);
```

$$A := \begin{bmatrix} 3 & 2 \\ 4 & 6 \end{bmatrix}$$

```
> B:=matrix(2,2,rand(20));
```

$$B := \begin{bmatrix} 1 & 10 \\ 17 & 3 \end{bmatrix}$$

```
> C:=diag(A,B);
```

$$C := \begin{bmatrix} 3 & 2 & 0 & 0 \\ 4 & 6 & 0 & 0 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 17 & 3 \end{bmatrix}$$

```
> C:=augment(A,B);
```

$$C := \begin{bmatrix} 3 & 2 & 1 & 10 \\ 4 & 6 & 17 & 3 \end{bmatrix}$$

La liste des fonctions mentionnées précédemment n'est pas exhaustive. Maple propose d'autres types de matrices spécifiques. Pour les découvrir, il est recommandé d'explorer l'ensemble des fonctions disponibles dans la bibliothèque "linalg".

17.4.2 Les fonctions de manipulation des matrices

- augment (A,B,...) crée une matrice en joignant horizontalement les matrices A,B,... C'est l'équivalent d'une concaténation des matrices.
- row(A,i) : extrait la ligne i de la matrice A.
- row(A,i..k) : extrait les lignes de i à k de la matrice A.
- col(A,i) extrait la colonne i de la matrice A.
- col(A,i..k) extrait les colonnes de i à k de la matrice A.
- delrows (A,r..s) supprime les lignes de r à s de la matrice A.
- delcols(A,r..s) supprime les colonnes r à s de la matrice A.
- copyinto(A,B,m,n) copie la matrice A dans la matrice B en plaçant l'élément A11 en Bmn et ainsi de suite.
- extend(A,m,n,x) ajoute m lignes et n colonnes initialisées à x à la matrice A
- minor(A,r,c) supprime la ligne r et la colonne c de la matrice A.
- submatrix(A,i..j,k..t) extrait la sous matrice de A constituée par les lignes i à j et les colonnes k à t.
- swapcol(A,c1,c2) permute les colonnes c1 et c2.
- swaprow(A,r1,r2) permute les lignes r1 et r2.

17.4.3 Effectuer des tests sur les matrices:

coldim(A) : retourne le nombre de colonnes de A.

rowdim(A) : retourne le nombre de lignes de A.

equal(A,B) : retourne TRUE si Aij=Bij pour tous i et j.

iszero(A) : retourne TRUE si Aij=0 pour tous i et j.

- colspace(A,'dim') retourne une base de l'espace vectoriel engendré par les vecteurs colonnes de la matrice. Le paramètre 'dim' est optionnel et définit la dimension de cet espace vectoriel.
- rowspace(A,'dim') retourne une base de l'espace vectoriel engendré par les vecteurs lignes de la matrice. 'dim' est la dimension de cet espace vectoriel.
- orthog(A) retourne TRUE si A est une matrice orthogonale, c-à-d, A est constituée de vecteurs colonnes ortho-normaux.
- rank(A) retourne le rang de la matrice A.
- trace(A) retourne la trace de la matrice A.

17.4.4 Opérations sur les matrices:

- Les opérateurs définis dans la section précédente diffèrent de ceux qui sont fournis dans le contexte du package « linalg ». Les opérateurs de « linalg » sont plus généraux et offrent plus de fonctionnalités.
- add(A,B,c1,c2) :permet de calculer la somme pondérée des matrices A et B à coefficients c1 et c2. N.B : cette fonction n'est pas disponible sur les anciennes versions de Maple.
- multiply(A,B,...) multiplie les matrices A,B,... entre-elles
- exponential(A) : retourne la matrice formée par l'exponentiel des éléments.
- addrow(A,li,lj) : crée une matrice A' où la ligne lj est remplacée par la somme de la ligne li et celle de la ligne lj.
- addcol(A,c1,c2) : crée une matrice A' où la colonne c2 est remplacée par la somme de c1 et c2.
- mulrow(A,r,expr) : multiplie la ligne r de la matrice A par expr.
- mulcol(A,c,expr) : multiplie la colonne c de la matrice A par expr.

- adjoint(A) : retourne la matrice adjointe de A. Elle est telle que A. A†=det(A)*I où I est la matrice identité. C'est équivalent à adj(A).
- htranspose(A) : retourne le transposé Hermitien de A.
- transpose(A) : retourne le transposé de la matrice A.
- det(A) : calcule le déterminant, s'il existe, de la matrice A.
- inverse(A) : calcule l'inverse, s'il existe, de la matrice carrée A.
- norm(A,opt) : calcule la norme de la matrice A suivant l'option spécifiée.
- charmat(A,lambda) : retourne la matrice caractéristique M de A.
- charpoly(A,lambda) : retourne le polynôme caractéristique P de A.
- eigenvals(A) : donne les valeurs propres de la matrice A. Ce sont aussi les solutions du polynôme caractéristique précédent.
- eigenvects(A) : renvoie les vecteurs propres de la matrice A sous la forme suivante : [[valeur propre i, multiplicité, {vecteur 1 associé, ...vecteur ni associé}...] où chaque vecteur j est associé à la valeur propre i et ni représente la dimension de l'espace propre correspondant à cette valeur propre.
- jordan(A) Maple renvoie la forme de Jordan de la matrice A, qui est une matrice diagonale où la diagonale est composée des valeurs propres de A.
- linsolve (A,B) si B est un vecteur , Maple fournit le vecteur x solution du système d'équations A.x=B. Si B est une matrice, Maple retourne la matrice X solution de l'équation matricielle A.X=B.

18 Référence :

- GARVAN, Frank. *The maple book*. CRC Press, 2001.
- CORLESS, Robert M. *Essential Maple: an introduction for scientific programmers*. Springer Science & Business Media, 2013.
- Thompson, Ian. *Understanding Maple*. Cambridge University Press, 2016.

19 Séries d'exercices

Exercice 1:

1. calculer la dérivée de la fonction f(x) = cos(x^2)+ln(x)/x^2.
2. calculer la somme : 1^3-2^3+3^3-… +n^3.
3. résoudre les équations :
   - $x^2 - 3y + 3 = 0$,
   - $-y^2 + 2x + 2 = 0$.

Exercice 2 :

3. Evaluer le cos de (3π)/5, en affichant 10 chiffres.
4. Evaluer sin(3π/7+5), en affichant 6 chiffres.

Exercice 3 :

Soit le nombre complexe z = (1+I)^2/(1-2*I).

1. Calculer la partie réelle Re(z)et la partie imaginaire Im(z)de z.
2. Calculer le module de z: abs(z).
3. Calculer l'argument de z: argument (z).
4. Calculer le conjuguais de z: conjugate(z).
5. Mettre z sous la forme a + iben utilisant la fonction : evalc().

Exercice 4 :

1. Stocker la constante "infini" ∞ dans la variable x et le nombre ln(2) dans la variable y.
2. Echanger les valeurs des variables x et y. par deux manières différentes
3. Vérifier les valeurs des variables x et y après l'échange.

4. Exécuter les instructions suivantes (la commande convert()) :
   - convert(123,binary);
   - convert(100,hex);
   - convert(101,decimal,binary);
   - convert(`1A`,decimal,hex);

5. Exécuter les instructions suivantes :
   - s :=a,b,c;
   - s :=s,d;
   - L :=[s];
   - nops(L);
   - op(L);
   - op(2,L);
   - L[3];
   - op(1..2,L);
   - L:=[op(L),e];
   - E :={1,2,3,2};
   - nops(E) ;
   - op(E);
   - op(2,E);
   - convert([op(E)],`*`);

Exercice 5 :

On part de deux variables

> **a :=2^100: b:=100 ! :**

b) Comment peut-on savoir si a > b ?

c) On souhaite échanger le contenu de ces deux variables. Comment s'y prendre ?

d) Calculer les facteurs premiers de ***a.***

e) Calculer le quotient et le reste de la division entière ***a/b***.

f) Calculer le PGCD et le PPCM des deux entiers ***a*** et ***b***.

g) Comment peut-on savoir si ***a+229*** est un nombre premier ?

h) Si oui, quel est le nombre premier suivant ?

i) Afficher le soixantième nombre premier.

Exercice 6 :

On considère les deux fonctions *f* et *g* définit sur [-5,5] par

$$f(x) = x^2 - x \text{ et } g(x) = 2x$$

1) Définir les fonctions *f* et*g#*
2) Résoudre la fonction *f(x) = 0# **solve(f(x) =0,x) ;***
3) Résoudre à l'aide de Maple les fonctions *f(x)=2, f(x)=g(x) ;*
4) Représenter graphiquement la fonction *f* pour *x* appartenant à [-5,5]
5) Tracer sur un même graphique les courbes représentatives de *f* et de ***g.***

Exercice 7 :

Soit f la fonction définie pour tout réel *x* >0 par

$$f(x) = x^{\left(\frac{x}{1-x}\right)}$$

1° Etablir les singularités de la fonction f et vérifier sa continuité.

2° Etudier les limites de f aux bornes des intervalles le domaine de définition.

3° Calculer f '(*x*) et étudier son signe (on aura recours à une fonction auxiliaire g).

4° Représenter la courbe de fonction *f*.

Exercice 8 :

1. afficher les éléments de la suite x^ipour ivariant de -5à 7(utiliser la commande seq et $).
2. Calculer le produit des termes de la suite x^ipour ivariant de -5à 7(utiliser lacommande convert).

Exercice 9 :

- Ecrivez une procédure qui calcule la somme des n premiers nombres entiers positifs. Vérifier le résultat en utilisant sum().

- Ecrivez une procédure qui calcule le factoriel de n, ou n est un entier positif. Vérifier le résultat en utilisant le factoriel : n!

Exercice 10:

1. Mettre dans une liste les valeurs du polynôme $x^2 + x + 41$pour xentier variant de -10à 10, (on définit le polynôme par p :=x->x^2+x+41).
2. Sélectionner les nombres premiers parmi les éléments de cette liste.

Exercice 11:

Exécuter les instructions suivantes :

1.
```
> for n from 5 to 10 do
        convert((n-1)^2,base,n);
        convert(2*(n-1),base,n) ;
  od;
```

2.
```
> for n from 5 to 10 do
        convert((n-1)^3,base,n);
        convert((n+2)*(n-1)^2,base,n) ;
  od;
```

3. Que constatez-vous ?

Exercice 12:

- On sait que l'état de l'eau dépend de sa température. Ecrire une procédure qui affiche l'état de l'eau (solide, liquide ou gazeux) selon sa température.

- Etant donnes deux nombres a et b, écrire une procédure qui nous informe si le signe du produit de a et b est négatif, positif ou nul (attention : on ne doit pas calculer le produit des deux nombres).

Exercice 13 :

Étant donnés deux matrices d'ordre 3 à coefficients réels suivantes :
A :=matrix(3,3,[-1,2,0,4,-2,3,0,1,-3]) et
B :=matrix(3,3,[2,0,1,4,-1,1,2,0,-5]).

1. Calculer la somme des deux matrices Aet B,
2. Calculer le produit du nombre -5par la matrice A,

3. Calculer la trace de la matrice Aet celle de la matrice B,
4. Calculer le produit de la matrice Apar la matrice B,
5. Calculer les déterminants det(A)et det(B).

Exercice 14 :

Ecrire un procédure Maple qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie :

- La catégorie d'un enfant est "Poussin" si $6 \leq age < 8$,
- La catégorie d'un enfant est "Pupille" si $8 \leq age < 10$,
- La catégorie d'un enfant est "Minime" si $10 \leq age < 12$,
- La catégorie d'un enfant est "Cadet" si $age \geq 12$.

Exercice 15 :

2. Ecrire une procédure Maple qui lit un entier n et compte le nombre de 1 dans la représentation binaire de cet entier.
3. Ecrire procédure maple qui lit un entier n (la taille du tableau), le tableau d'entiers T den éléments, l'entier x et indique le nombre de fois que x figure dans le tableau T.
4. Ecrire une procédure maple qui prend pour arguments un entier n (la taille du tableau), letableau de réels T et affiche le plus grand élément du tableau T ainsi que sa positiondans le tableau.