INTRODUCTION

À

MAPLE

Par

Guy Jomphe

TABLE DES MATIÈRES

Qu'est-ce que Maple ?

Maple est un logiciel de mathématiques développé par Waterloo **Maple** Software. Il se distingue par la puissance de son calcul symbolique, numérique et par la représentation graphique des résultats.

C'est un logiciel destiné aux scientifiques, ingénieurs, étudiants possédant un bon niveau mathématique. Ce logiciel fait des merveilles dans le calcul à très haute précision, la résolution d'équations réelles, imaginaires, différentielles, intégrales, etc.

La prise en main du logiciel **Maple** est relativement aisée, pourvu que l'on maîtrise les mathématiques. Néanmoins **Maple** possède un didacticiel complet.

Le module graphique de base de **Maple** permet le changement de couleur discontinu et les changements dynamiques des points de vue de graphiques, Cette dernière caractéristique pouvant être très intéressante pour visualiser les surfaces et volumes en 3D.

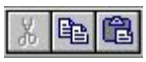
Interface de Maple

En ouvrant le logiciel Maple, il apparaît à l'écran un interface graphique dans lequel on retrouve une barre de menu dans la partie supérieure de l'écran et une feuille de travail, dans la partie centrale. Cette feuille de travail appelée 'worksheet' en anglais est un gros espace blanc et commencant par l'invite [>. Ce dernier symbole signifie que le logiciel est en attente d'instructions. Plusieurs feuilles peuvent être ouvertes en même temps.

La barre d'outil principale du logiciel, voir ci-dessous, est constituée de différents boutons permettant de manipuler la feuille de travail.

Description des boutons

Les quatre premiers boutons permettent, dans l'ordre, d'ouvrir une nouvelle feuille de travail vierge, d'ouvrir une feuille déjà existante, de sauver cette feuille, et d'imprimer la feuille de travail.



Les trois boutons suivants permettent de couper, de copier et coller certaines parties de la feuille de travail.

Ces deux boutons servent à défaire ou refaire les dernières actions.

Ces trois boutons suivants servent respectivement à insérer des expressions mathématiques, du texte et des lignes.

1. Le bouton avec un **T** permet l'insertion de texte à la suite de la position du curseur.
2. Le bouton avec un imprimé **Σ** permet d'entrer des expression mathématiques qui seront converties sous forme littérale.
3. Le troisième bouton permet l'insertion de lignes. Cet ajout se fait en positionnant le curseur à la fin de la ligne courante et en cliquant ensuite sur le bouton **[>**. L'insertion de la ligne se fera immédiatement après la ligne courante. En répétant le processus, il est possible d'ajouter autant de lignes que nécessaire.

Prendre note que les symboles [>, [ou [? apparaissant en début de ligne indiquent dans quel mode vous vous situez : en attente d'instruction Maple, de texte ou de conversion.

Les deux boutons suivants servent à hiérarchiser les paragraphes dans la feuille de travail.

Ce bouton-ci avec l'imprimé **STOP** sert à arrêter les calculs mathématiques en cours.

Ces trois boutons servent à agrandir le contenu de la feuille de travail à 100 %, 150 % ou 200 %.

GÉNÉRALITÉS

Syntaxe

Comme n'importe quel langage de programmation, **Maple** possède sa propre syntaxe et la connaissance de celle-ci est primordiale.

Opérateurs de base:

+ / -	addition / soustraction
*	multiplication
^	puissance (symbole synonyme: **)
/	division

Fonctions usuelles:

abs (z)	:	valeur absolue d'un nombre ou module d'un nombre complexe z
sqrt(z)	:	racine carrée de z
exp(z)	:	exponentielle de z
ln(z)	:	logarithme népérien de z

Fonctions trigonométriques et hyperboliques directes:

sin(z), cos(z), tan(z), sec(z), csc(z), cot(z)

sinh(z), cosh(z), tanh(z), sech(z), csch(z), coth(z)

et inverses:

arcsin, arccos, arctan, arcsec, arccsc, arccot

arcsinh, arccosh, arctanh, arcsech, arccsch, arccoth

Symboles particuliers:

I	:	nombre imaginaire tel que $\sqrt{-1} = I$
Pi	:	valeur numérique : 3.1415.....
pi	:	lette grecque π (aucune valeur numérique)

rem: 1.

[> sin(pi); **Maple** retournera sin(π) et non 0

[> sin(Pi) **Maple** retournera 0

2.

Maple " parle " grec. En ce sens qu'une lettre grecque peut être entrée à l'écran en tapant les lettres à l'aide du clavier. **Maple** la convertira sous forme symbolique.

Ex:

[> sin(beta) ; **Maple** retournera sin(ß)

Commandes de base:

Description	*Exemple*	*Sortie Maple*	*Commentaires*	*Note*
;	**[> 3+4 ;**	7	Exécute et affiche le résultat	
:	**[> 3+4 :**		Exécute la commande mais n'affiche pas le résultat	
:=	**[> a := cos(5*Pi);** **[> b := a / 2.5;**	a := -1 b:= -.4	C'est ce que l'on appelle une affectation	
evalf	**[>A:=evalf(7/3);** **[>A:= evalf(7/3, 4);**	A :=2.3333 A :=2.333	Évalue sous forme décimale Évalue avec 4 chiffres	
restart	**[> restart;**		re-initialise la feuille de travail	Se place au début de la feuille de travail.
simplify	**[>simplify(cos(ß^2 +sin(ß^2);**	1	Pour faire des simplifications	
rhs	**[>eq:=3*sin(ß)=cos(ß):** **[> rhs(eq);**	cos(ß)	rhs : abréviation du mot anglais right-hand- side	Permet de prendre le membre de droite d'une expression
lhs	**[>eq:=3*sin(ß)=cos(ß):** **[> lhs(eq);**	3 sin(ß)	lhs : abréviation du mot anglais left-hand- side	Permet de prendre le membre de gauche d'une expression
solve	**[>solution:=solve(3*x=6, x);**	solution:= x=2	Pour résoudre une équation	
int	**[>res:=int(sin(x), x=-Pi..Pi);**	res:= 0	Pour intégrer	
diff	**[>dérivée:= diff(y(x),x,x);**	dérivée:= $\frac{\partial^2}{\partial x^2} y(x)$	Dérivée seconde par rapport à x de y(x)	
subs	**[>subs(x=2,sin(x*a));**	sin(2 a)	Substitution de x=2 dans sin(x a)	
plot	**[>plot(f(x), x=a..b):**		Dessinera la fonction f(x) sur [a,b]	Commande valable en 2-D seulement
assume	**[>assume(n,integer)**		Sert à faire des hypothèses.	

Notions de liste, ensemble

Ensemble: collection **non-ordonnée** et sans répétition d'objets. Les accolades { } sont utilisées pour les dénoter. Un ensemble se forme en plaçant des objets entre accolades.

Ex:

```
[> Ens := { 5 , 4, 4, 7  , sin(x) };
                                        Ens := {4,5,7 , sin(x) }
```

Liste: collection **ordonnée** d'objets. Les répétitions sont conservées. Les crochets [ ] sont utilisés pour les dénoter, c'est-à-dire que des objets entre crochets est une liste et qu'on forme une liste en plaçant des objets entre crochets.

Ex:

```
[> liste :=[ 5 , 4, 4, 7  , sin(x) ];          ( Conserve l'ordre des éléments)
                                        liste := [ 5 , 4 , 4 ,  7  , sin(x) ]
```

Suite: collection d'objets séparés par des virgules

Ex :

```
[> suite :=  3,4,5,6,7 ;
                                        suite :=  3,4,5,6,7
```

Pour accéder aux éléments d'un ensemble ou d'une liste, on utilise les crochets [ ]

Ex :

```
[>F:={1, 3, 3, 2, 4, 4, 2, sin(x)};
                                        F:={1, 2, 3, 4, sin(x)};
```

Attention: Dans ce dernier exemple, il se peut que l'ordre des éléments dans l'ensemble soit changé par **Maple**.

```
[> F[5];
                                        sin(x)
```

Pour enlever les { } ou les [ ] d'un ensemble ou d'une liste ,on aura alors une suite, il suffit d'utiliser les [ ] n'ayant aucun "argument".

Ex : [>F:={12 , 3 , 3 , 2 , 4 , 4 , 2};

```
[>F;
                                        {2, 3, 4, 12 }
[>f[];
                                        2, 3, 4, 12
```

Création d'une fonction mathématique

Le logiciel possède déjà des fonctions pré-définies comme cos, sin, ln,…. Mais il est possible de créer sa propre fonction mathématique

Ex : Soit la fonction f : x |-----> $\cos(x) + \sin(x^2)$. Alors il est possible de trouver f (a).

[> f := x -> cos(x)+sin(x^2) : L'utilisation de **:** empêche l'affichage du résultat

[>f(a);

rép : cos(a)+sin(a^2)

de même:

[> f (x+y);

rép : cos(x+y)+sin((x+y)^2)

Pour convertir une expression en une fonction, on peut utiliser la commande **unapply** .

[> p := x^3 + 1;

rép : p:=x^3+1

[>fonction:=unapply(p,x);

rép : fonction :=x---------> x^3+1

[>fonction(a);

rép : a^3+1

Les modules ("packages")

Lorsque vous ouvrez le logiciel **Maple**, plusieurs commandes de base (evalf, simplify,..) peuvent être utilisées immédiatement. Mais pour des commandes plus spécialisées dans des domaines précis tels: l'Algèbre linéaire , Équations différentielles, etc, il est nécessaire d'introduire le module approprié à l'aide de la commande **with**.

Ainsi pour travailler avec les matrices et les vecteurs, il est nécessaire d'introduire le module d'Algèbre linéaire du nom de *linalg* et cela, préférablement après la commande **restart**.

[> restart;

[>with (linalg);

Cette dernière commande affichera une série de commande pour traiter un grand nombre de notion d'Algèbre linéaire. Si vous ne voulez pas l'affichage de toutes les commandes, remplacer **;** par **:** dans with(linalg);

À titre d'exemple, voici certains modules avec le sujet traité:

Plots :	Représentation graphiques
Linalg :	Algèbre linéaire
Detools :	Outils pour les Équations Différentielles
Powseries :	Séries formelles
inttrans :	Transformation intégrales : Laplace, Fourier
stats :	Statistiques

Pour la liste complète des modules, taper: [> ?index, package;

Premiers pas avec Maple

À l'invite [> de la feuille de travail, taper **restart;** pour annuler tous les résultats précédents. Ne pas oublier de faire un retour de chariot pour permettre l'exécution de la commande et cela pour chaque ligne de commandes.

```
[>restart;
```

Q1

Trouver la valeur de 1/4 - 3/8

```
[>1/4 - 3/8;
                    -1/8
```

Donner le résultat sous forme décimale.

```
[> evalf (1/4-3/8);     Placer le curseur devant 1/4- 3/8  et taper evalf plutôt que de re-
                        taper  l'expression. )
                    -.1250000000
```

Q2

Écrire l'expression : $\cos(a\ x) + \tan(x^2) - \sin(5\pi) + \frac{I}{2} + e^{2\ x}$ et lui affecter le nom de expression.

```
[> expression:= cos(a*x)+tan(x^2) -sin(5* Pi) + I /2 + exp(2* x);
```

expression := $\cos(a\ x) + \tan(x^2) - \sin(5\pi) + \frac{I}{2} + e^{2x)}$

Q3

En utilisant la commande **diff**, trouver la dérivée première par rapport à x de $\cos(a\ x) + e^{2\ x}$ et lui affecter le nom de dérivée.

```
[> expr:=cos(a*x) + exp(2*x);
                    expr:=cos(a x) + exp(2x)
[> dérivée:=diff(expr, x);
                    dérivée:= -a sin(a x)+2 exp(2 x)
```

Q4

En utilisant la commande **int**, trouver la valeur de l'intégrale entre 0 et π /2 de la fonction $\cos(\ x)\ \ e^{2\ x}$.

```
[>fonction:= cos(x)*exp(x);
[> Valeur_de l'_intégrale:= int (fonction, x = 0..Pi/2);
```

Valeur de l'intégrale:= $\frac{1}{2} \quad e^{\pi/2} - \frac{1}{2}$

[> evalf(%); Le % fait référence à la dernière commande

1.905238691

Q5

En utilisant la commande **subs**, remplacer la variable a par 2 dans l'expression tan(2 a x) + sin(ß)

[>expr:= tan(2* a* x) + sin(beta);

expr:= tan(2 a x) + sin(ß)

[> résultat:= subs(a=2, expr); (attention, ne pas écrire 2 = a)

résultat:= tan(4 x) + sin(ß)

Q6

À l'aide de la commande **plot** et **rhs** , faire dessiner la fonction $y = x^2$ entre 0 et 3.

[>equation:= y=x^2;

equation:= $y = x^2$

[> plot(rhs(equation), x = 0..3);

Maple retournera le graphique de la fonction x^2

Q7

Créer une liste contenant les éléments 4, 4, 2, sin(x) et lui donner le nom de liste

[> liste:= [4, 4 , 2, sin(x)]; Les accolades carrées [ ] signifient que nous avons une liste et donc tous les éléments à l'intérieur seront conservés . De même que l'ordre lors de la sortie du résultat.)

liste:= [4 , 4 , 2 , sin(x)]

Q8

Créer un ensemble contenant les éléments 4, 4, 2, sin(x) et lui donner le nom de ensemble.

[>ensemble:= {4, 4, 2, sin(x) }; Les accolades { } signifient que les éléments à l'intérieur qui se répètent seront enlevés et que l'ordre ne sera pas conservé lors de la sortie du résultat.)

ensemble:= { 2, 4, sin(x) }

Q9

Résoudre l' équation sin(x) = x

[>équation:= sin(x) = x ;

équation:= sin(x) = x

[> solve(équation, x);

0

[> solve(équation, { x }); L'utilisation de { } permettent d'associer le nom de la variable à sa valeur.

{ x = 0 }

Q10

Créer une suite dont les éléments sont 4, sin (x), cos(x), 8, 9 et de cette suite, retirer le 2 ième élément sin(x).

```
[> suite:= 4, sin (x),  cos(x), 8 , 9  ;
```

suite := 4, sin (x), cos(x), 8 , 9

```
[> suite[2];
```

[2] permet de soutirer le 2 ième élément parmis la suite.)

sin (x)

autre possibilitée:

```
[> suite[- 4];
```

Le - indique que le comptage se fait à partir de la fin de la suite)

sin (x)

GRAPHISME 2-D

Pour dessiner des courbes en deux dimensions, plusieurs commandes sont disponibles. Le choix de l'une de ces commandes se fait selon que la fonction est exprimée sous forme: explicite, implicite, paramétrique ou sous forme polaire.

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 à dessiner la fonction qui est donnée sous la forme implicite f (x, y) = 0

Pour cela, il est nécessaire d'introduire le module *plots*. Ainsi :

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

Ex :

Dessiner la courbe dont l'équation cartésienne est $e^{xy} = \cos(x - y)$ pour $0 \le x \le \pi$ *et* $2 \le y \le 4$

Noter que y = y (x).

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

Forme paramétrique

Soit à tracer une fonction sur l'intervalle [a, b] et dont les équations paramétriques sont:

x = f (t)

y = g (t)

La syntaxe de la commande est

[> 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); Ceci introduit le module *plots*

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

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

GRAPHISME 3-D

Forme explicite

Soit à dessiner une surface dont l'équation de cette surface est donnée sous forme

z = f (x , y).

La syntaxe de la commande est :

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

Ex :

Soit à dessiner la surface $z = x^2 + y^2$ pour $-1 \le x \le 1$ et $-2 \le y \le 2$

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

Forme implicite

Soit à dessiner la surface S lorsque l'équation de cette surface est donnée sous forme

F(x ,y , z) =0

Les commandes sont:

```
[>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) $a \le t \le b$

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) $0 \le t \le 2\pi$

z = t

Les commandes sont:

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

Soit à dessiner une surface dont on connaît les équations paramétriques suivantes:

x = f (t , s)

y = g (t , s) $a \le t \le b \quad et \quad c \le s \le d$

z = h (t , s)

La commande est

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

Graphisme en coordonnées cylindriques et sphériques

Lorsque l'équation de la surface à dessiner est exprimée en coordonnées :cylindriques, ou sphériques, on peut aussi utiliser la commande *plot3d* avec l'option *coords* qui peut prendre l'une des valeurs cylindrical, ou spherical'' selon que l'on utilise les coordonnées cylindriques ou sphériques respectivement.

En coordonnées cylindriques

La distance r, entre un point d'une surface et l'axe z est exprimé par :

$$r = r(\theta, z) \quad \text{avec} \quad \theta_1 \le \theta \le \theta_2 \quad \text{et} \quad z_1 \le z \le z_2$$

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 $z^2 = x^2 + y^2$ 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 :

$$\rho = \rho(\theta, \phi) \quad \text{avec} \quad \theta_1 \le \theta \le \theta_2 \quad \text{et} \quad \phi_1 \le \phi \le \phi_2$$

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 );
```

Dessiner des courbes de niveau

Soit à dessiner le diagramme des courbes de niveau d'une fonction $f(x,y)$ pour c = c1,…, cn.

Ex $f(x,y) = \dfrac{-4}{1+x^2+y^2}$ pour c= -1, -2, -3, -4.

[>restart;

[>with(plots);

[>c1 :=-1; Déclaration des constantes c

[>c2 :=-2;

[>c3 :=-3;

[>c4 :=-4;

[>fonction := $\dfrac{-4}{1+x^2+y^2}$

Pour afficher la surface $z = f(x,y)$ à l'écran.(facultatif)

[>smartplot3d(fonction);

Pour visualiser en 3D les courbes d'intersection entre la surface z=f(x,y) et le plan z = c. (facultatif)

[>contourplot3d (fonction, x = -10..10, y = -10..10);

Pour visualiser en 2D les courbes de niveau dans le plan xy.

[>smartplot(fonction = c1);
[>smartplot(fonction = c1, fonction = c2,);
[>smartplot(fonction = c1, fonction = c2, fonction = c3);
[>smartplot(fonction = c1, fonction = c2, fonction = c3, fonction = c4);

Rem : Pour modifier les paramètres : échelle, couleur, etc, d'un graphique, il suffit de cliquer sur celui-ci pour voir apparaître une barre d'outil par laquelle on peut accéder aux paramètres.

2. Pour modifier l'échelle en particulier, cliquez sur le graphique puis dans la barre d'outil, cliquez sur **Projection** ensuite sélectionnez **Constrained**.

Dérivation

Dérivation explicite

Commande: diff ou Diff Commande active ou inerte

Syntaxe:

diff(expression, x1, x2, x3,....., xn);

diff(expression, x1, x2, x3,....., xn);

Ex :

Soit à calculer $\frac{\partial}{\partial x} f(x, y)$ où f(x,y) = sin(x+y) ln(x y)

[>expression:= sin(x+y)*ln(x *y);

[>diff (expression, x);

Ex :

Soit à calculer $\frac{\partial^2}{\partial x \partial y} f(x, y)$ où f(x, y) = sin(x + y) ln (x y)

[>expression:= sin(x + y)*ln(x *y);

[>diff (expression, y, x);

Dérivation implicite

Commande: implicitdiff Dérivation d'une fonction définie par une équation.

Syntaxe:

implicitdiff(équation, fonction dépendante, variable de dérivation);

Ex :

Soit à trouver dy/dx de cos(x + y) = ln(x y) où y = y(x).

[>expression:= cos(x+y) = ln(x* y);

[>implicitdiff (expression, y, x);

Dérivation en chaîne

Lors de la dérivation en chaîne, il est impératif d'identifier les variables dépendantes et indépendantes.

Ex :

Soit à retrouver la formule $\frac{dz}{dt} = \frac{dz}{dx}\frac{dx}{dt}$ où z= z(x) et x = x(t).

[>restart;

[>g := z(x(t)); Ne pas écrire z (x)

[>Diff (g, t))= diff(g, t);

Le résultat faisant apparaître l'opérateur de dérivation D, il est possible de convertir (facultatif) celui-ci en terme de notation habituelle utilisant le symbole ∂.

[>convert (%, diff);

Ex :

Soit à trouver $\frac{dz}{dt}$ en utilisant la règle de la dérivée en chaîne. pour la fonction $z = xy^2$

$$\text{où } \begin{cases} x = e^{-t} \\ y = \sin(t) \end{cases}.$$

[>restart;

[>f := $z(x(t), y(t)) = x(t) * y(t)^2$; Il faut définir explicitement de quelles variables dépendent x et y.

[>D_selon_t := diff (f , t); Calcule la dérivée de f par rapport à t

[>var := x(t) =exp(-t), y(t)=sin(t); Changement de variables

[>subs (var, D_selon_t);

[>diff(z(t),t) = eval(%);

Trouvez $\frac{\partial^2 f(x,y)}{\partial u \partial v}$ où x = x(u, v) et y = y (u, v).

[>restart;

[>g :=f(x(u, v), y(u, v));

[>Diff(g, v, u)= diff (g ,v, u);

INTÉGRATION

Intégrale simple

Commandes: int ou Int Commande active et inerte respectivement

Syntaxes:

int(f(x), x);

Int(f(x), x);

int(f(x), x=a..b); Évaluera l'intégrale et affichera le résultat.

Int(f(x), x=a..b); Retournera l'intégrale non-évaluée

Paramètres:

f(x) fonction à intégrer

x variable d'intégration

a, b bornes d'intégration inférieure et supérieure respectivement

Ex :

Soit à intégrer la fonction f(x) = sin(x)

[> int(sin(x), x);

Ex :

Soit à intégrer la fonction f(x) = sin(x) sur [-π, π].

[> int (sin(x), x = -Pi..Pi);

Intégrale double

Commandes: Doubleint

1.Double étant une commande inerte, pour obtenir l'évaluation de l'intégrale double, il faut utiliser soient: eval, evalf,evalc,etc.

2. Pour utiliser cette commande, il est nécessaire d'ouvrir le module student à l'aide de with(student). Voir exemple ci dessous

Syntaxes:

Doubleint (f(x, y), x, y) L'intégrale se fera selon x et ensuite selon y

Doubleint (f(x , y), x =a..b, y = c..d)

Doubleint (f(x , y), x, y, Domaine)

Paramètres:

f(x,y) fonction ou expression à intégrer

a, b, c, d bornes d'intégration

Domaine Nom du domaine qui apparaîtra sous les intégrales

Ex :

Soit à intégrer la fonction f(x, y) = x + y sur le domaine D = [1,2]x[3,4]

[>with(student); Introduction du module *student*

[>Doubleint (x + y, x =1..2, y = 3..4);

[>evalf (%); Le signe % fait référence à la dernière commande

NOTIONS VECTORIELLES

Vecteur

Nous pouvons utiliser la notion de liste, par l'utilisation de [ ], pour représenter des points ou des vecteurs.

Ainsi les points P(a, b, c) et Q(c, d,e) s'écriront:

[> P := [a, b, c];

[> Q := [c, d, e];

[> PQ := Q-P;

Maple retournera le vecteur PQ:= [c- a, d- b, e- c]

rem :

Il est impératif d'utiliser des crochets [...] plutôt que des parenthèses (...) ou des accolades {...}, car l' ordre des éléments a, b, ..., e sera ainsi respecté.

De même pour écrire les vecteurs :

$\vec{V} = v1\,\vec{i} + v2\,\vec{j} + v3\,\vec{k}$ et

$\vec{W} = w1\,\vec{i} + w2\,\vec{j} + w3\,\vec{k}$

[> V := [v1, v2, v3];

[> W := [w1, w2, w3];

Pour des vecteurs entrés sous la forme V := [v1, v2, v3] , **Maple** ne fait pas de distinction entre des vecteurs lignes et des vecteurs colonnes. L'interprétation est laissée à l'utilisateur. Pour plus de rigueur mathématique, il serait préférable d'écrire les vecteurs sous forme matricielle. Rendant ainsi plus aisée les opérations matricielles.

Ex : au lieu d'écrire : [> V := [v1, v2, v3];

écrire plutôt [> V := matrix (3, 1, [v1, v2, v3]); matrice de 3 lignes et 1 colonne

Afin de traiter les problèmes d'Algèbre linéaire, il est nécessaire d'introduire le module d'Algèbre linéaire du nom de *linalg* au tout début de la feuille de travail, immédiatement après la commande restart:

```
[>restart;
[>with (linalg);   Cette commande donne accès à plus de 200 autres commandes
[>                 Désormais les commandes du module linalg peuvent être utilisées.
```

Produit vectoriel et scalaire de 2 vecteurs

Soit $\vec{V} = v1\,\vec{i} + v2\,\vec{j} + v3\,\vec{k}$ et

$\vec{W} = w1\,\vec{i} + w2\,\vec{j} + w3\,\vec{k}$

```
[> restart;
[> with ( linalg);          Permet d'utiliser des commandes qui sont à l'intérieur du module linalg
[>V := [v1, v2, v3];
[>W := [w1, w2, w3];
[>innerprod ( V, W);        Produit scalaire dans une base orthonormale
[>crossprod( V, W);         Produit vectoriel dans une base orthonormale
[>angle_entre_V_W := angle( V, W);      Calcule la valeur de l'angle entre V et W
[>norme_de_V:= sqrt (innerprod (V,V));  Calcule la norme du vecteur V :
```

$$\left\|\vec{V}\right\| = \sqrt{v1^2 + v2^2 + v3^2}$$

Dérivation vectorielle

Soit le vecteur $\vec{r}(t) = a\cos(t)\,\vec{i} + a\sin(t)\,\vec{j} + c\,t\,\vec{k}$

Calculer la dérivée première, deuxième, troisième et dixième de $\vec{r}(t)$ par rapport à t.

```
[restart;
[>r:= [a*cos (t), a*sin (t), c*t];
[> r_prime:= diff(r, t);          Calcule la dérivée première
[> r_seconde:= diff( r, t ,t );   Calcule la dérivée seconde
[> r_troisième:= diff( r, t ,t ,t );   Calcule la dérivée troisième
[> r_dixième:= diff( r, t $10 );   Calcule la dérivée dixième
```

Gradient, Laplacien, Divergence, Rotationnel

Soit f(x, y, z) une fonction scalaire et

$\vec{g}(x, y, z) = g1(x, y, z)\,\vec{i} + g2(x, y, z)\,\vec{j} + g3(x, y, z)\,\vec{k}$ une fonction vectorielle

Alors:

```
[restart;
[>with(linalg);
[>fonction_f := f(x, y, z );
[> fonction_g := [ g1(x, y, z ),  g2(x, y, z),   g3(x, y, z) ];
```

Calcul du gradient de la fonction f(x, y, z):

```
[>  gradient:=  grad (f(x, y, z ), [x, y, z ] );
```

Calcule les dérivées premières, dans l'ordre de f(x, y, z) par rapport à x, y, z.

Calcul du Laplacien de la fonction f(x, y, z):

```
[> lap_de_f:= laplacian (fonction_f, [x, y, z ] );
```

Calcule les dérivées deuxième, dans l'ordre, de f(x, y, z) par rapport à x, y, z.

Calcul du rotationnel de la fonction g(x, y, z):

```
[>rot_de_g := curl( fonction_g, [x,y,z] );
```

Il faut écrire les composantes de $\vec{g}$ sous forme de liste.

Calcul de la divergence de la fonction g(x, y, z):

```
[>div_de_g := divergence( fonction_g,  [x, y, z] );
```

DÉVELOPPEMENT EN SÉRIE DE TAYLOR

Série de Taylor

Commande: taylor

Syntaxes:

taylor (f(x), x =a, n);

Paramètres:

f(x)	Fonction dont il faut écrire le développement en série de Taylor
a	Point autour duquel f(x) est développé en série de Taylor
n	Ordre du développement

Ex :

Soit à développer en série de Taylor autour du point x = 0 de f(x) = sin(x)

[>série_Taylor := taylor(sin(x), x = 0, 3);

série_Taylor := x + $O(x^3)$)

Polynôme de Taylor

De la série de Taylor, il est possible d'extraire le polynôme de Taylor à l'aide de la commande **convert** et de l'option *polynom.*

Ex :

[>série_Taylor := taylor(sin(x), x = 0, 3);

série_Taylor := x + $O(x^3)$)

[> poly := convert(série_Taylor, polynom);

poly := x

Pour les séries entières on peut utiliser le module *powseries* qui contient plusieurs commandes permettant de manipuler ces séries.

[>restart;

[>with(powseries);

[>.....

Extrémums d' une fonction

Trois commandes de base sont disponibles pour trouver les extrémums d'une fonction d'une ou de plusieurs variables :

Extrema (expression, {contrainte}, {variable})

Minimize (expression, option1, option2,..., option3)

Maximize (expression, option1, option2,..., option3)

Ex: Trouver le minumum de $(x-2)^2$ sur l' intervalle [1, 5]

[> restart;

[> fonction:= (x-2)^2;

[> minimize(fonction, x = 1..5, location); location indiquera quelles sont les coordonnées du point où se trouve ce minimum.

Ex: Trouver le minimum et le maximum de $f(x,y) = x^2 + y^2$ sur le domaine [-3,3] X [-4..4]

```
[> restart;
[> fonction:=  x^2 +y^2;
[>  minimize( fonction, x = -3..3, y = -4..4, location );
[>  maximize( fonction, x = -3..3, y = -4..4, location );
```

Ex:

Soit une sphère centrée à l'origine de rayon 2 et dont l'équation est $x^2 + y^2 + z^2 = 2$. Sachant que la température sur la sphère est donnée par la fonction $T(x,y,z) = x^2 + y^2 + z^2$, trouver les extrémums de T(x, y, z).

```
[> restart;
[> fonction:=  x^2 +y^2+   z^2;
[> extrema (fonction, x^2+y^2+z^2=2, { x, y, z } );
```

ÉQUATIONS DIFFÉRENTIELLES

Soit à résoudre une équation différentielle ayant une ou plusieurs conditions initiales. Cette résolution se fait en utilisant la commande

dsolve({eq_diff, init, cond_init } , {y(x)})

où eq_diff : Équation différentielle

init : Condition(s) initiale(s)

y(x) : Fonction inconnue à rechercher

Ex:

Résoudre y''(x) + y'(x) + y(x) = 3 e^x avec les conditions initiales: y(0) =1, y'(0)=0

```
[> restart;
[> eq_diff := diff (y(x), x, x) + diff (y(x), x ) + y(x)  = 3*exp(x);
[> cond_init  :=  y(0) = 0, D(y)(0) = 0;
[> solution := dsolve( {eq_diff,init, cond_init } , { y(x)} );
```

y(x): = fonction en x.

```
[> plot (solution, x = 0..12) ) ;
```

Faire attention , comme la fonction à dessiner se trouve du côté droit du signe de l'égalité, il faut utiliser la commande **rhs**

Note: Si la fonction se serait située à gauche du signe de l'égalité, nous aurions utilisé la commande **lhs.**

```
[> plot ( rhs( solution), x = 0..12) ;
```

Le graphe n'a pas de titre ni de nom donné aux axes.

```
[> plot ( rhs( solution), x = 0..12, title = " Graphe de la solution ", labels = [ x, " y(x)"] ) ;
```

Le graphe aura un titre et un nom

Champ de directions

Nous nous servons de la notion de champ de directions pour approximer la solution d'une équation différentielle du premier ordre de la forme:

$$y'(x) = f(x, y(x))$$

Cette méthode du champ de direction consiste à dessiner l'allure du comportement de la solution y(x) en se basant qu'en tous points (xo, yo), la pente de la solution y(x) en ce point est f (xo, yo).

Il est à noter que cette méthode est particulièrement intéressante lorsque la fonction f (x, y(x)) n'est pas intégrable.

Ex:

Tracer le champ de direction de la fonction y'(x) = e^{x^2} ayant comme condition initiale y(0) = 1.

```
[> restart;
[> with ( DEtools);
```

Introduction du module *DEtools* afin de pouvoir utiliser divers outils pour traiter les équations différentielles. En autre, la commande **DEplot.** pour dessiner la solution.

```
[> eq_diff :=  diff(y(x),x) = exp(x^2);
[> cond_init := y(0)=1;
[> Deplot ( eq_diff, y(x), x = -2..4, [[ cond_init]], y = -3..5, stepsize = 0.05,  title = " Champ de
  directions ");
```

ALGÈBRE LINÉAIRE

Pour traiter les problèmes d'Algèbre linéaire, il est nécessaire d'introduire un module du nom de *linalg* et cela , de la façon suivante:

```
[>  restart;
[> with( linalg);   200 commandes sont maintenant disponibles
```

Ex :

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Calculer le déterminant de la matrice A, son inverse de même que ses valeurs propres et vecteurs propres.

```
[>restart;
[> with(linalg);
```

[> A := matrix(2, 2, [a, b, c, d]); Définie une matrice 2 lignes par 2 colonnes et possédant les éléments a, b, c, d

[> det (A); Calcul le déterminant de la matrice A

[>B := inverse(A); Calcul l'inverse de A et lui donne le nom de matrice B

[>multiply(A, B); Effectue la multiplication des matrice A et B

[> eigenvalues(A); Retourne une suite valeurs propres

[> vect :=eigenvectors(A); Retournera une suite de liste suivante :

vect :=[vap#1, multiplicité de vap#1,{vep#1}], [vap#2, multiplicité de vap#2,{vep#2}]

Il est possible d'extraire vap#2 et vep#2, de la façon suivante:

[> vect[2];

[vap#2, multiplicité de vap#2,{vep#2}]

[> vect[2,1];

vap#2 La 2 ième valeur propre

[> vect[2,3];

{vep#2} Le 2 ième vecteur propre

DIVERS

#1 Extraction d'informations

Soit la séquence de 2 listes suivante:

[> rest := [a, b, c] , [23, sin(x), { cos(x),tan(x)] ;

Pour extraire la lettre c:

[>rest[1, 3]; Le chiffre 1 fait référence à la première liste [a,b,c] et le 3 au 3 ième élément à l'intérieur de cette liste [a,b,c]

Pour extraire la fonction tan(x):

[>rest[2, 3, 2];

#2 Graphisme

Soit à tracer une courbe mais Maple me retourne empty plot , plotting error

[> restart;

[> fonction:= y(x) = sin(x);

[> plot(fonction, x = a..b); La fonction à tracer :sin(x), étant à droite du signe de l' égalité, il faut donc prendre le côté doit de l' équation à l' aide de la commande **rhs**. Ainsi

[> plot(rhs(fonction), x = a..b);

#3 Indice

Soit à écrire la lettre B avec un indice j en bas à droite.

[>restart;

[> B[j] ;

Maple retournera B_j

Mise à jour: 18 juillet 2001