Mémento Maple V Release 5

***

I. Généralités 2

1°) Environnement 2
2°) Affectation 2
3°) Programmation 2

II. Nombres et fonctions 3

1°) Nombres entiers, arithmétique 3
2°) Nombres rationnels 3
3°) Nombres flottants 3
4°) Nombres complexes 3
5°) Fonctions reconnues par Maple 3

III. Les objets Maple, les objets mathématiques 4

1°) Les applications 4
2°) Les polynômes 4
3°) Les fractions rationnelles 4
4°) Séquences, listes et ensembles 5

IV. Analyse 5

1°) Calculs analytiques 5
2°) Représentations graphiques 6
3°) Résolutions formelles et numériques d'équations 7
4°) Transformation d'expressions 8
5°) Manipulation d'égalités et d'inégalités 9

V. Algèbre linéaire 9

1°) Calcul vectoriel 9
2°) Calcul matriciel 9
3°) Géométrie vectorielle euclidienne 10

VI. Raccourcis clavier utiles 10

1°) Raccourcis standards Windows 10
2°) Raccourcis particuliers à Maple 11

VII. Ecran Maple 11

I. Généralités

1°) Environnement

restart L'instruction **restart** réalise une remise à zéro du système.

with **with(nomDuPackage)** charge en mémoire le contenu du package concerné.

I,Pi symbolisent respectivement les constantes $i=\sqrt{-1}$ et **Erreur ! Des objets ne peuvent pas être créés à partir des codes de champs de mise en forme.**

infinity correspond au symbole $+\infty$, **-infinity** correspond à $-\infty$.

? **?instruction** donne accès à l'aide concernant l'instruction voulue.
??instruction donne accès à la syntaxe de cette instruction.
???instruction donne directement accès aux exemples d'utilisation.

% Le pourcentage **%** remplace le dernier résultat calculé par Maple, le double pourcentage **%%** remplace l'avant dernier résultat et le triple pourcentage **%%%** remplace l'antépénultième.

print **print(expr1,expr2,...)** entraîne l'affichage des valeurs des expressions expr1, expr2,...
Pour afficher du texte (formaté), on utilisera l'instruction **printf**.

2°) Affectation

assign **assign(a,expr)** ou **a:=expr** réalise l'affectation a ← expr de l'algorithmique.
assign(a=expr) transforme l'égalité **a=expr** en affectation **a:=expr**.

assume **assume(a,prop1,prop2,...)** affecte les propriétés **prop1, prop2,...** à la variable **a**.
Cette variable est alors notée $a\sim$, le tilde soulignant qu'une propriété lui est attribuée.
Parmi les propriétés que l'on peut attribuer citons : **complex**, **real**, **rationnal**, **integer**, **odd**, **even**, **natural**, **prime**, **mapping**, **scalar**, **vector**, **matrix**, **squarematrix**,...
assume(inégalité1,inégalité2,...) affirme que la variable concernée satisfait les inégalités : **inégalité1, inégalité2,...**

about **about(a)** rappelle l'ensemble des propriétés attribuées à **a**.

unassign **unassign('a')** ou **a :='a'** supprime l'affectation ou les propriétés attribuées à **a**.

is **is(x,prop)** teste si **x** satisfait à la propriété **prop**.
is(inégalité) teste si l'inégalité concernée est vérifiée.
Ex : **assume(x,real,x<>0);is(x^2>0);** *true*

3°) Programmation

proc Une procédure **nomProc** fonction des arguments **para1**, **para2**,... et renvoyant le résultat **resultat** se définit de la manière suivante :

```
nomProc:=proc(para1,para2,...)
global variable_globale1,variable_globale2,... ;
local variable_locale1,variable_locale2,... ;
instruction1;
instruction2;
...;
RETURN(resultat);
end;
```

On notera l'absence de point-virgule à la première ligne. La deuxième et la troisième lignes, définissant les variables globales et locales sont optionnelles.

if La construction **if...then...fi;** traduit le 'si ... alors ... finsi' de l'algorithmique.
La construction **if...then...else...fi;** traduit 'si ... alors ... sinon ... finsi'.
La construction **if...then...elif...then...elif...then... ...fi;** permet d'emboîter rapidement les instructions conditionnées.
Il ne faut pas oublier le **fi** final.

for La construction **for i from i1 to i2 do...od;** traduit le 'pour i allant de i1 à i2 ... finpour' de l'algorithmique.
La construction **for i from i1 by pas to i2 do...od;** permet de spécifier une

incrémentation de **i** valant **pas**.
La construction **for i in {...} do ... od;** fait décrire à i les éléments de l'ensemble proposé.

while La construction **while...do...od;** traduit le 'tant que ... faire ... fintantque' de l'algorithmique.

II. Nombres et fonctions

1°) Nombres entiers, arithmétique

factorial **factorial(n)** ou **n!** calcule le produit factoriel de **n**.

binomial **binomial(n,p)** calcule le coefficient binomial **p** parmi **n**.

convert **convert(expr,factorial)** réécrit expr en exploitant la notation factorielle.

iquo **iquo(a,b)** retourne le quotient de la division euclidienne de **a** par **b**.

irem **irem(a,b)** retourne le reste de la division euclidienne de **a** par **b**.

mod **a mod b** retourne le reste de la division euclidienne de **a** par **b**.

igcd **igcd(a,b,...)** retourne le pgcd des entiers **a**, **b**,...

ilcm **ilcm(a,b,...)** retourne le ppcm des entiers **a**, **b**,...

ifactor **ifactor(n)** donne la décomposition primaire de n.

isprime **isprime(p)** teste si l'entier p est premier ou non.

ithprime **ithprime(i)** retourne le $\mathbf{i}^{\text{ème}}$ nombre premier.

2°) Nombres rationnels

numer **numer(r)** retourne le numérateur du représentant irréductible de **r**.

denom **denom(r)** retourne le dénominateur du représentant irréductible de **r**.

3°) Nombres flottants

evalf **evalf(x)** détermine une approximation à **Digits** chiffres du nombre **x**.
evalf(x,n) détermine une approximation à **n** chiffres du nombre **x**.

convert **convert(x,rational)** écrit un nombre flottant **x** sous forme rationnelle.

4°) Nombres complexes

evalc **evalc(z)** écrit le nombre complexe z sous forme algébrique.

Re,Im **Re(z)** et **Im(z)** retournent la partie réelle et la partie imaginaire du complexe **z**.

abs **abs(z)** retourne le module de **z**.

argument **argument(z)** détermine l'argument de **z** appartenant à $]-\pi,\pi]$.

conjugate **conjugate(z)** retourne le conjugué de **z**.

polar **polar(r,theta)** définit le complexe de module **r** et d'argument **theta**.

convert **convert(z,polar)** écrit le complexe **z** sous la forme **polar(r,theta)**.

5°) Fonctions reconnues par Maple

abs	valeur absolue	**floor**	partie entière	**sqrt**	racine carrée
exp	exponentielle	**ln**	logarithme népérien	**log[a]**	logarithme en base **a**.
sin,cos,tan,cot			fonctions trigonométriques.		
arcsin,arccos,arctan			fonctions trigonométriques inverses.		
sinh,cosh,tanh,coth			fonctions trigonométriques hyperboliques.		
arcsinh,arccosh,arctanh			fonctions trigonométriques hyperboliques inverses.		

GAMMA fonction Γ d'Euler définie par $\Gamma(x)=\int_0^{+\infty} t^{x-1}\mathrm{e}^{-t}\,\mathrm{d}t$.

III. Les objets Maple, les objets mathématiques

1°) Les applications

unapply **f:=unapply(expr,x)** définit l'application qui à **x** associe **expr**. Dans le cas où **expr** dépendrait explicitement de **x**, on peut plus intuitivement écrire **f:=x->expr**.
f:=unapply(expr,x1,x2,...) définit la fonction qui à **(x1,x2,...)** associe **expr**.
Comme ci-dessus on peut encore écrire **f:=(x1,x2,...)->expr**.

piecewise **f:=x->piecewise(cond1,expr1,cond2,expr2,...,expr_sinon)** définit l'application qui à **x** associe **expr1** si **cond1** est remplie, **expr2** si **cond2** est remplie,... et **expr_sinon** si aucune des conditions **cond1**, **cond2** n'est satisfaite.

eval **eval(f)** force l'évaluation complète de la fonction **f**.

@ L'arobase **@** (s'obtient par [AltGr] + [0]) symbolise le produit de composition des applications.

@@ La double arobase **@@** permet de composer plusieurs fois une application avec elle-même, ainsi **(f@@k)** correspond à $f \circ f \circ ... \circ f$ (**k** termes).

D **D(f)** correspond à la dérivée de l'application **f**, **f** est ici une fonction d'une variable.
(D(D(f)) ou **(D@D)(f)** correspond à la dérivée seconde de l'application **f**.
Plus généralement, **(D@@k)(f)** est la dérivée **k**ème de **f**.

D[i](f) et **D[i1,i2,...ip](f)** correspondent aux dérivées partielles $\frac{\partial f}{\partial x_i}$, $\frac{\partial^p f}{\partial x_{i1}\partial x_{i2}...\partial x_{ip}}$.

2°) Les polynômes

degree **degree(P,x)** donne le degré du polynôme **P** en l'indéterminée **x**.

ldegree **ldegree(P,x)** donne la valuation du polynôme **P** en l'indéterminée **x**.

coeff **coeff(P,x^k)** ou **coeff(P,x,k)** donne le coefficient du terme **x^k** dans le polynôme **P**.

lcoeff **lcoeff(P,x)** donne le coefficient dominant du polynôme **P** en l'indéterminée **x**.

sort **sort(P,x)** réécrit le polynôme **P** selon les puissances décroissantes de **x**.

collect **collect(P,x)** regroupe les monômes du polynôme **P** en l'indéterminée **x**.

solve **solve(P,x)** fournit la séquence des racines du polynôme **P** en l'indéterminée **x**.

roots **roots(P)** fournit la liste des racines du polynôme **P** ainsi que leur multiplicité.
roots(P,{α1,α2,...}) autorise Maple à chercher les racines de **P** sur un corps contenant les nombres algébriques **α1**, **α2**,...

Ex : **roots(4*x^3-8*x^2-3*x+9);** $[[\frac{3}{2},2],[-1,1]]$

factor **factor(P)** factorise le polynôme **P**.
factor(P,{α1,α2,...}) autorise Maple à factorise le polynôme **P** sur un corps contenant les nombres algébriques **α1**, **α2**,...

Ex : **factor(x^3-x^2+2*x-2);** $(x-1)(x^2+2)$

factor(x^3-x^2+2*x-2,{sqrt(2),I}); $(x+I\sqrt{2})(x-I\sqrt{2})(x-1)$

quo,rem **quo(P,Q)** et **rem(P,Q)** calculent quotient et reste de la division euclidienne de **P** par **Q**.

gcd,lcm **gcd(P,Q)** et **lcm(P,Q)** calculent un pgcd et un ppcm des polynômes **P** et **Q**.

3°) Les fractions rationnelles

normal **normal(F)** écrit la fraction rationnelle **F** sous forme irréductible.
L'option **expanded** permet, de plus, de développer le dénominateur

numer **numer(F)** donne le numérateur de **F**.

denom **denom(F)** donne le dénominateur de **F**.

convert **convert(F,parfrac,x)** décompose en éléments simples la fraction rationnelle **F** en l'indéterminée **x**. **convert(F,parfrac,x,{α1,α2,...})** autorise Maple à travailler sur un corps contenant les nombres algébriques **α1**, **α2**,...

4°) Séquences, listes et ensembles

seq **seq(f(i),i=i1..i2)** ou **f(i) $ i=i1..i2** génère une séquence de la forme : **f(i1),f(i1+1),...,f(i2)**.
Ex : **seq(i^2,i=1..10);** 1, 4, 9, 16, 25, 36, 49, 64, 81, 100

NULL **NULL** symbolise la séquence vide.

min **min(S)** détermine le plus petit élément de la séquence **S**.

max **max(S)** détermine le plus grand élément de la séquence **S**.

union **E union F** calcule la réunion des ensembles **E** et **F**.

intersect **E intersect F** calcule l'intersection de **E** et **F**.

minus **E minus F** calcule l'ensemble formé par **E** privé de **F**.

convert **convert(E,list)** transforme l'ensemble **E** en une liste.
convert(L,set) transforme la liste **L** en un ensemble.

nops **nops(E)** détermine le nombre d'éléments constituant l'ensemble **E**.
nops(L) détermine le nombre d'éléments constituant la liste **L**.

op **op(k,L)** ou **L[k]** donne le **k**ème élément de la liste **L**.
op(k,E) ou **E[k]** donne le **k**ème élément de l'ensemble **E**, **E** étant arbitrairement ordonné.
op(A) transforme la liste ou l'ensemble **A** en une séquence.

member **member(x,A)** teste si l'élément **x** appartient ou non à la liste ou l'ensemble **A**.

map **map(f,A)** applique la fonction **f** à chacun des termes de la liste ou de l'ensemble **A**.
Ex : **map(i->ithprime(i),{seq(i^2,i=1..10)});**
{2, 7, 23, 53, 97, 151, 227, 311, 419, 541}

select **select(f,A)** recherche parmi les éléments de la liste ou de l'ensemble **A**, ceux qui vérifient la fonction à valeurs booléennes **f**.
Ex : **select(x->isprime(x),[seq(i,i=1000..1020)]);** [1009,1013,1019]

IV. Analyse

1°) Calculs analytiques

diff **diff(expr,x)** calcule la dérivée de **expr** par rapport à la variable **x**.
diff(expr,x,x,...) ou **diff(expr,x,y,...)** donne les dérivées d'ordres supérieurs.

int **int(expr,x)** primitive **expr** par rapport à la variable **x**.
int(expr,x=a..b) calcule l'intégrale de **expr** pour **x** variant de **a** à **b**.

limit **limit(expr,x=a)** retourne la limite de **expr** quand **x** tend vers **a**.
Les options **left**, **right**, **real** précisent la direction dans laquelle **x** tend vers **a**.

sum **sum(expr,i=i1..i2)** calcule la somme des termes **expr** pour **i** allant de **i1** à **i2**.

product **product(expr,i=i1..i2)** calcule le produit des termes **expr** pour **i** allant de **i1** à **i2**.

Les instructions ci-dessus possèdent une forme inerte : **Diff**, **Int**, **Limit**, **Sum**, **Product**. Ces formes inertes peuvent être évaluées par l'instruction **value**.

Ex : **Int(exp(-x^2),x=0..infinity):"=value(");** $\int_0^{\infty} e^{-x^2} dx = \sqrt{\pi}/2$

series **series(expr,x=a)** calcule un développement asymptotique de **expr** au voisinage de **a**.
series(expr,x=a,n) calcule ce développement en manipulant des développements à **n** termes. La valeur par défaut est **n=Order=5**.

taylor **taylor(expr,x=a)** calcule le développement de Taylor de **expr** au voisinage de **a**.
taylor(expr,x=a,n) calcule ce développement à l'ordre **n**. Par défaut **n=Order=5**.

convert **convert(expr,polynom)** tronque la partie infinitésimale d'un développement asymptotique.

Ex : **convert(taylor(exp(x),x=0),polynom);** $1+x+\frac{1}{2}x^2+\frac{1}{6}x^3+\frac{1}{24}x^4+\frac{1}{120}x^5$

Les deux instructions suivantes appartiennent au package **student**.

intparts **intparts(Int(f(x),x=a..b),u(x))** réalise une intégration par partie en dérivant le terme **u(x)** à l'intérieur de l'intégrale.

changevar **changevar(eqn,Int(f(x),x=a..b),u)** réalise le changement de variable transformant **x** en **u** à l'intérieur de l'intégrale, **x** et **u** étant liant par l'équation **eqn**.

2°) Représentations graphiques

smartplot **smartplot(f(x))** crée une représentation graphique de la courbe d'équation **y=f(x)** pour **x** variant de –10 à 10. En cliquant sur le bouton droit de la souris on peut modifier différents paramètres de la représentation

plot **plot(f(x),x=a..b)** représente la courbe d'équation **y=f(x)** pour **x** variant de **a** à **b**.

plot(f(x),x=a..b,y=c..d) représente la courbe **y=f(x)** dans la fenêtre **[a,b]×[c,d]**.
plot([f(x),g(x),...],x=a..b) représente simultanément **y=f(x)**, **y=g(x)**,...
plot([x(t),y(t),t=t1..t2]) représente la courbe paramétrée $t \mapsto M(t) = (x(t), y(t))$.
plot([[x0,y0],…,[xn,yn]]) représente une ligne brisée dont les sommets sont les points spécifiés
L'option **color=red**, **blue**, **green**, **black**, **pink**, **gold**... spécifie la couleur du tracé.
L'option **thickness=0**, **1**, **2**, ... spécifie l'épaisseur du tracé croissant avec l'entier précisé.
L'option **discont=true** évite les barres verticales lors de la représentation de fonctions présentant des discontinuités.
L'option **scaling=constrained** impose une représentation dans un repère orthonormé.
L'option **view=[x1..x2,y1..y2]** réalise une représentation dans **[x1,x2]×[y1,y2]**.
L'option **coords=polar** spécifie que l'on travaille en coordonnées polaires.
L'option **title="..."** permet de donner un titre au graphique.

Ex 1 :
```
plot(tan(x),x=-Pi..Pi,y=-5..5,
     discont=true,scaling=constrained);'
```
Ex 2 :
```
plot([sin(x)+sin(7*x)/7,sin(x)],x=-Pi..Pi,
     thickness=[0,2],color=[blue,red]);
```

plot3d **plot3d(f(x,y),x=x1..x2,y=y1..y2)** représente la surface d'équation **z=f(x,y)**
plot3d([x(u,v),y(u,v),z(u,v),u=u1..u2,v=v1..v2]) représente la nappe paramétrée $(u,v) \mapsto M(u,v) = (x(u,v), y(u,v), z(u,v))$.

L'option **view=z1..z2** ou **view=[x1..x2,y1..y2,z1..z2]** impose une représentation dans le pavé **[x1,x2]×[y1,y2]×[z1,z2]**.
Les options **coords=cylindrical** ou **coords=spherical** spécifient que l'on travaille en coordonnées cylindriques ou sphériques.

Enfin signalons l'existence de l'instruction **smartplot3d** qui prolonge l'instruction **smartplot** aux représentations graphiques en dimension 3.

Ex 3 :
```
plot3d(z^2,theta=0..2*Pi,z=-1..1,coords=cylindrical);
```

Les instructions qui suivent appartiennent au package **plots**.
Les options de l'instruction **plot** sont aussi disponibles pour la plupart des instructions suivantes.

display **display([G1,G2,...])** affiche les graphiques **G1, G2,...** simultanément.
L'option **insequence=true** transforme la série de graphique **G1,G2,...** en animation.

Ex4 :
```
A:=plot(cos(x),x=0..Pi):B:=plot(arccos(x),x=-1..1):
display([A,B]);
```

pointplot **pointplot([[x1,y1],[x2,y2],...])** représente les points de coordonnées **(x1,y1)**, **(x2,y2)**, ... L'option **connect=true** trace une ligne entre chacun de ces points. On peut aussi utiliser l'instruction **plot** pour parvenir à ce résultat.

Ex 5 :
```
liste:=[seq([n,ithprime(n)],n=1..50)]:
pointplot(liste,connect=true) ;
```

polarplot **polarplot(r(t),t=t1..t2)** représente la courbe d'équation polaire $\rho = r(t)$. On peut aussi utiliser l'instruction **plot** avec l'option **coords=polar** pour obtenir le même résultat.

cylinderplot, sphereplot permettent de représenter une surface définies en coordonnées cylindriques ou sphériques.

implicitplot **implicitplot(f(x,y)=0,x=x1..x2,y=y1..y2)** trace la courbe d'équation cartésienne **f(x,y)=0** dans la fenêtre **[x1,x2]×[y1,y2]**.

animate **animate(F(x,t),x=a..b,t=t1..t2)** forme une animation en représentant le graphe de **x ↦ F(x,t)** et en faisant varier **t** depuis **t1** jusqu'à **t2**.

L'option **frames=n** spécifie le nombre **n** de graphes constituant l'animation.

La valeur par défaut est 16.

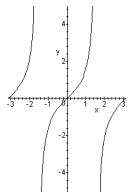
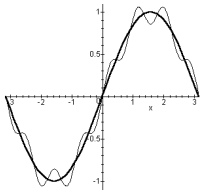
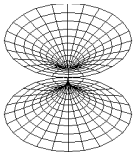
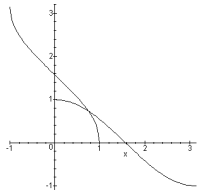
Ex : **animate(1+a*cos(t),t=0..2*Pi,a=0..2,coords=polar,frames=30);**

animate3d **animate3d(f(x,y,t),x=x1..x2,y=y1..y2,t=t1..t2)** construit une animation en dimension 3.

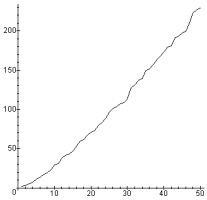
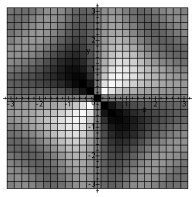
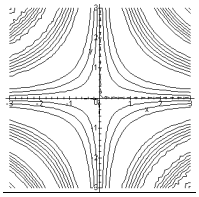
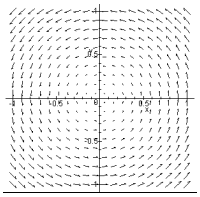
Citons enfin les instructions **densityplot**, **contourplot** et **fieldplot** pour tracer des graphes de densité, des lignes de niveaux et des champs de vecteurs.

Ex 6 : **densityplot(sin(x*y)/(x^2+y^2),x=-3..3,y=-3..3);**
Ex 7 : **contourplot(sin(x*y),x=-3..3,y=-3..3);**
Ex 8 : **fieldplot([-y,x],x=-1..1,y=-1..1);**

Ex 1 Ex 2 Ex 3 Ex 4

Ex 5 Ex 6 Ex 7 Ex 8

3°) Résolutions formelles et numériques d'équations

RootOf **RootOf(eqn,x)** symbolise une des solutions de **eqn** en l'inconnue **x**.
Pour obtenir toutes les valeurs possibles il faut utiliser l'instruction **allvalues**.
Si Maple ne peut déterminer de solutions exactes, il donnera des valeurs approchées.

Ex : **RootOf(z^3+z^2+z+1);** $\mathrm{RootOf}(_Z^3+_Z^2+_Z+1)$
allvalues("); $-1, I, -I$

solve **solve(eqn,x)** résout formellement l'équation **eqn** en l'inconnue **x**.
solve({eqn1,eqn2,...},{x1,x2,...}) résout formellement le système formé par les équations **eqn1**, **eqn2**,... en les inconnues **x1**,**x2**,...

Ex : **solve(z^2+z=5*(1-I),z);** $2-I, -3+I$

Les solutions générées par l'instruction **solve** peuvent parfois faire usage de **RootOf**.

Ex : **eqns:={x+y=1,x^2+y^2=2};** $eqns := \{x+y=1, x^2+y^2=2\}$
solve(eqns,{x,y});
$$\{y=\mathrm{RootOf}(2_Z^2-2_Z-1), x=-\mathrm{RootOf}(2_Z^2-2_Z-1)+1\}$$
allvalues(%); $\{x=\frac{1}{2}-\frac{1}{2}\sqrt{3}, y=\frac{1}{2}+\frac{1}{2}\sqrt{3}\}, \{x=\frac{1}{2}+\frac{1}{2}\sqrt{3}, y=\frac{1}{2}-\frac{1}{2}\sqrt{3}\}$

fsolve **fsolve(eqn,x)** résout numériquement l'équation **eqn** en l'inconnue **x**.
L'option **x=a..b** spécifie une recherche de solutions dans l'intervalle **[a,b]**.
L'option **complex** demande une recherche de solutions dans le plan complexe.

fsolve({eqn1,eqn2,...},{x1,x2,...}) résout numériquement le système formé par les équations **eqn1**, **eqn2**,... en les inconnues **x1**,**x2**,...
L'option **x1=a1..b1**, **x2=a2..b2**, ... spécifie les intervalles de recherche.

dsolve **dsolve(eqDiff,y(x))** résout formellement l'équation différentielle **eqDiff** de fonction inconnue **x ↦ y(x)**.

Les constantes $_C1, _C2, \ldots$ susceptibles d'apparaître sont les constantes d'intégration.

Ex : **dsolve(D(D(y))(x)+omega^2*y(x)=0,y(x));** $y(x) = _C1\cos(\omega x) + _C2\sin(\omega x)$

dsolve({eq_diff,cond1,cond2,...},y(x)) résout formellement le problème différentielle formé par l'équation différentielle **eq_diff** et les conditions **cond1**, **cond2**,...

Ex : **dsolve({D(D(y))(x)-y(x)=0,y(0)=1,D(y)(0)=1},y(x)) ;** $y(x) = e^x$

dsolve({eq_diff1,eq_diff2,...,cond1,cond2},{y1(x),y2(x),...} permet de résoudre des problèmes différentiels faisant intervenir plusieurs fonctions inconnues.
L'option **type=numeric** résout numériquement le problème posé.
Maple retourne alors une procédure *proc(rkf45_x) ... end* qui donne une approximation numérique des valeurs prises par la solution du problème différentiel posé.

Ex : **f:=dsolve({D(D(y))(x)=sin(y(x)),y(0)=1,D(y)(0)=0},y(x), type=numeric);** $f := \mathrm{proc}(rkf45_x)\ldots\mathrm{end}$

f(1); $[x = 1, y(x) = 1.437288371053132, \frac{\partial}{\partial x}y(x) = .9024307411105019]$

rsolve **rsolve(relation_rec,u(n))** détermine le terme général **u(n)** des suites satisfaisant à la relation de récurrence **relation_rec**.
rsolve({relation_rec,cond1,cond2,...},u(n)) détermine de plus les suites satisfaisant aux conditions **cond1**, **cond2**,...
rsolve({relation_rec1,relation_rec2,...,cond1,cond2,...},{u1(n), u2(n),...}) permet de résoudre des problèmes faisant intervenir plusieurs suites inconnues

Ex : **rsolve({u(n+1)=u(n)+4*v(n),v(n+1)=u(n)+v(n),u(0)=1,v(0)=0}, {u(n),v(n)});** $\{v(n) = -\frac{1}{4}(-1)^n + \frac{1}{4}3^n, u(n) = \frac{1}{2}(-1)^n + \frac{1}{2}3^n\}$

Citons aussi l'instruction **isolve** qui résout les équations dont les inconnues sont des entiers relatifs.

4°) Transformation d'expressions

eval **eval(expr)** force l'évaluation de l'expression **expr**.
A l'inverse si l'expression est mise entre apostrophe : **'expr'**, son évaluation sera retardée.
eval(expr,x=a) réalise l'évaluation de **expr** en donnant la valeur **a** à la variable **x**.

expand **expand(expr)** développe l'expression **expr** en distribuant les produits sur les sommes mais aussi en exploitant les relations : $\cos(a+b) = \ldots, \exp(a+b) = \ldots$, etc...

combine **combine(expr)** transforme **expr** en essayant de recombiner plusieurs termes en un seul.

factor **factor(expr)** factorise l'expression **expr**.

normal **normal(expr)** réduit les termes de **expr** au même dénominateur.

simplify **simplify(expr)** transforme l'expression **expr** en y appliquant des règles de simplification.
simplify(expr,{ident1,ident2,...} simplifie **expr** en y appliquant de plus les relations spécifiées dans les identités **ident1**, **ident2**,...

Ex : **simplify(a^3+b^3+3*a*b,{a+b=1});** 1

Les options **ln**, **exp**, **power** (fonctions puissances), **trig** (fonctions trigonométriques), **sqrt** (racine carrée), **radical** (puissances rationnelles) spécifient de n'appliquer que les règles de simplifications relatives aux fonctions concernées.
L'option **assume=real** spécifie que chacune des variables rencontrées dans **expr** est réelle.
Maple peut alors appliquer à **expr** les simplifications usuelles sur les réelles.
L'option **assume=postive** spécifie que chacune des variables rencontrées est réelle positive.
Maple peut alors appliquer certaines simplifications.

Ex : **test:=exp(cos(x)^2)*exp(sin(x)^2);** $test := e^{(\cos^2(x))}e^{(\sin^2(x))}$

```
simplify(test),simplify(test,exp),simplify(test,trig);
```
$$e, e^{(\cos^2(x)+\sin^2(x))}, e^{(\cos^2(x))}e^{(1-\cos^2(x))}$$

subs **subs(x=a,expr)** substitue toutes les occurrences de **x** en **a** dans **expr**.
subs({x=a,y=b,...},expr) effectue simultanément les substitutions **x** → **a**, **y** → **b**,...

Ex : `subs(x=Pi/4,cos(x)+sin(x));eval(");` $\cos\left(\frac{1}{4}\pi\right)+\sin\left(\frac{1}{4}\pi\right)$ $\sqrt{2}$

Ex : `subs({x=y,y=x},y*exp(x));` xe^{y}

whattype **whattype(expr)** fournit le type de l'expression **expr**.
Ex : `whattype(a+b+c);whattype(cos(x));` $+$ *function*

Citons encore les instructions **nops**, **op**, **convert**, **map** et **select** qui peuvent opérer sur les expressions tout comme elles opéraient sur les ensembles ou les listes.

5°) Manipulation d'égalités et d'inégalités

lhs,rhs **lhs(relation)** et **rhs(relation)** fournissent le membre de gauche et de droite de l'égalité ou de l'inégalité **relation**.

isolate **isolate(relation,x)** isole le terme **x** à l'intérieur de l'égalité ou de l'inégalité **relation**. Cette instruction appartient au package **student**.

assign **assign(a=expr)** transforme l'égalité **a=expr** en affectation **a:=expr**.

V. Algèbre linéaire

1°) Calcul vectoriel

Les instructions suivantes ne sont disponibles qu'à partir du package **linalg**.

vector **vector([a1,a2,...,an])** définit le vecteur de composantes **(a1,a2,..., an)**.
Matriciellement, un vecteur se comporte comme une ligne ou une colonne.
Si **v** est un vecteur, **v[i]** correspond à la **i** ème composante du vecteur **v**.

equal **equal(v,w)** teste si les vecteurs v et **w** sont égaux.

basis **basis([v1,v2,...])** donne une base de l'espace vectoriel engendré par les **v1**, **v2**,...

intbasis **intbasis([v1,v2,...],[w1,w2,...])** détermine une base de l'intersection des sous-espaces vectoriels engendrés par **(v1,v2,...)** et **(w1,w2,...)**.

sumbasis **sumbasis([v1,v2,...],[w1,w2,...])** détermine une base de la somme des sous-espaces vectoriels engendrés par **(v1,v2,...)** et **(w1,w2,...)**.

Ex :
```
v1:=vector([1,0,2]);v2:=vector([1,2,3]);
w1:=vector([1,2,0]);w2:=vector([-1,0,1]);
```
$v1 := [1,0,2]$ $v2 := [1,2,3]$ $w1 := [1,2,0]$ $w2 := [-1,0,1]$
```
sumbasis({v1,v2},{w1,w2});intbasis({v1,v2},{w1,w2});
```
$\{w1, w2, v2\}$ $\{[0,2,1]\}$

2°) Calcul matriciel

Les instructions suivantes ne sont disponibles qu'à partir du package **linalg**.

matrix **matrix([[a11,a12,...,a1p],[a21,a22,...,a2p],..,[an1,an2,...,anp]])** définit la matrice de type **(n,p)** et de coefficients $(a_{i,j})_{1\le i\le n, 1\le j\le p}$.

matrix([v1,v2,...,vn]) définit la matrice dont les lignes sont constituées par les vecteurs **v1**,**v2**,...,**vn**.

matrix(n,p,f) définit la matrice de type **(n,p)** et de coefficients $(f(i,j))_{1\le i\le n, 1\le j\le p}$ où **f** est une fonction de deux variables.

Si **A** est une matrice **A[i,j]** correspond au coefficient d'indice **(i,j)** de **A**.

Ex : `A:=matrix(2,3,(i,j)->i*j);A[1,3];` $A := \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \end{pmatrix}$ 3

concat **concat(v1,v2,...,vp)** forme la matrice dont les colonnes sont les vecteurs **v1**, **v2**, ...,**vp**.

diag **diag(a1,a2,...an)** forme la matrice diagonale de coefficients **a1**, **a2**,..., **an**.

evalm **evalm(A)** demande une évaluation de **A** sous forme matricielle.

equal **equal(A,B)** teste si les matrices **A** et **B** sont égales.

iszero **iszero(A)** teste si la matrice **A** est nulle.

transpose **transpose(A)** calcule la matrice transposée de **A**.

trace **trace(A)** calcule la trace de la matrice carrée **A**.

matadd **matadd(A,B)** ou **evalm(A+B)** calcule la somme matriciel **A+B**.

scalarmul **scalarmul(A,lambda)** ou **evalm(lambda*A)** multiplie **A** par le scalaire **lambda**.

multiply **multiply(A,B)** ou **evalm(A&*B)** calcule le produit matriciel **A×B**.

rank **rank(A)** calcule le rang de la matrice **A**.

inverse **inverse(A)** ou **evalm(A^(-1))** calcule l'inverse de la matrice carrée **A**.

det **det(A)** calcule le déterminant de la matrice carrée **A**.
Ex : **A:=matrix(2,2,[a,b,c,d]):det(A);** $ad-bc$

kernel **kernel(A)** donne une base du noyau de l'application linéaire associée à **A**.

colspace **colspace(A)** donne une base de l'espace vectoriel engendré par les colonnes de **A**.
On obtient ainsi une base de l'image de **A**.

rowspace **rowspace(A)** donne une base de l'espace vectoriel engendré par les lignes de **A**.

eigenvals **eigenvals(A)** détermine les valeurs propres de la matrice **A**.

eigenvects **eigenvects(A)** forme une séquence de listes constituées des valeurs propres de **A**, de leur multiplicité et d'une base du sous-espace propre associé.

charpoly **charpoly(A,X)** donne le polynôme caractéristique de **A** en l'indéterminée **X**.

minpoly **minpoly(A,X)** donne le polynôme minimal de **A** en l'indéterminée **X**.

3°) Géométrie vectorielle euclidienne

Les instructions suivantes ne sont disponibles qu'à partir du package **linalg**.

dotprod **dotprod(v1,v2)** calcule le produit scalaire/hermitien des vecteurs **v1** et **v2**.

norm **norm(v)** donne la norme euclidienne/hermitienne du vecteur **v**.

crossprod **crossprod(v1,v2)** calcule le produit vectoriel des vecteurs **v1** et **v2**.

GramSchmidt **GramSchmidt([v1,v2,....])** retourne la base orthogonale obtenue à partir de la famille de vecteurs **(v1,v2,...)** par le procédé de Gram-Schmidt.

orthog **orthog(A)** teste si la matrice **A** est orthogonale ou non.

VI. Raccourcis clavier utiles

1°) Raccourcis standards Windows

Déplacement du point d'insertion

[↖] positionne le curseur en début de ligne.

[Fin] positionne le curseur en fin de ligne.

[Ctrl] + [↖] positionne le curseur au début du plan de travail.

[Ctrl] + [Fin] positionne le curseur à la fin du plan de travail.

[Ctrl] + Déplacement curseur : Déplacement rapide du point d'insertion.

[⇑] + [Entrée] : passe le curseur à la ligne

Technique du copier/coller

[⇑] + Déplacement du curseur : sélection de texte.

Click gauche + Déplacement souris : sélection de texte.

[Ctrl] + [A] sélectionne tout.

[Ctrl] + [C] copie le texte sélectionné en mémoire (clipboard).

[Ctrl] + [X] coupe le texte sélectionné et le place en mémoire.
[Ctrl] + [V] colle le texte qui a précédemment été mis en mémoire.

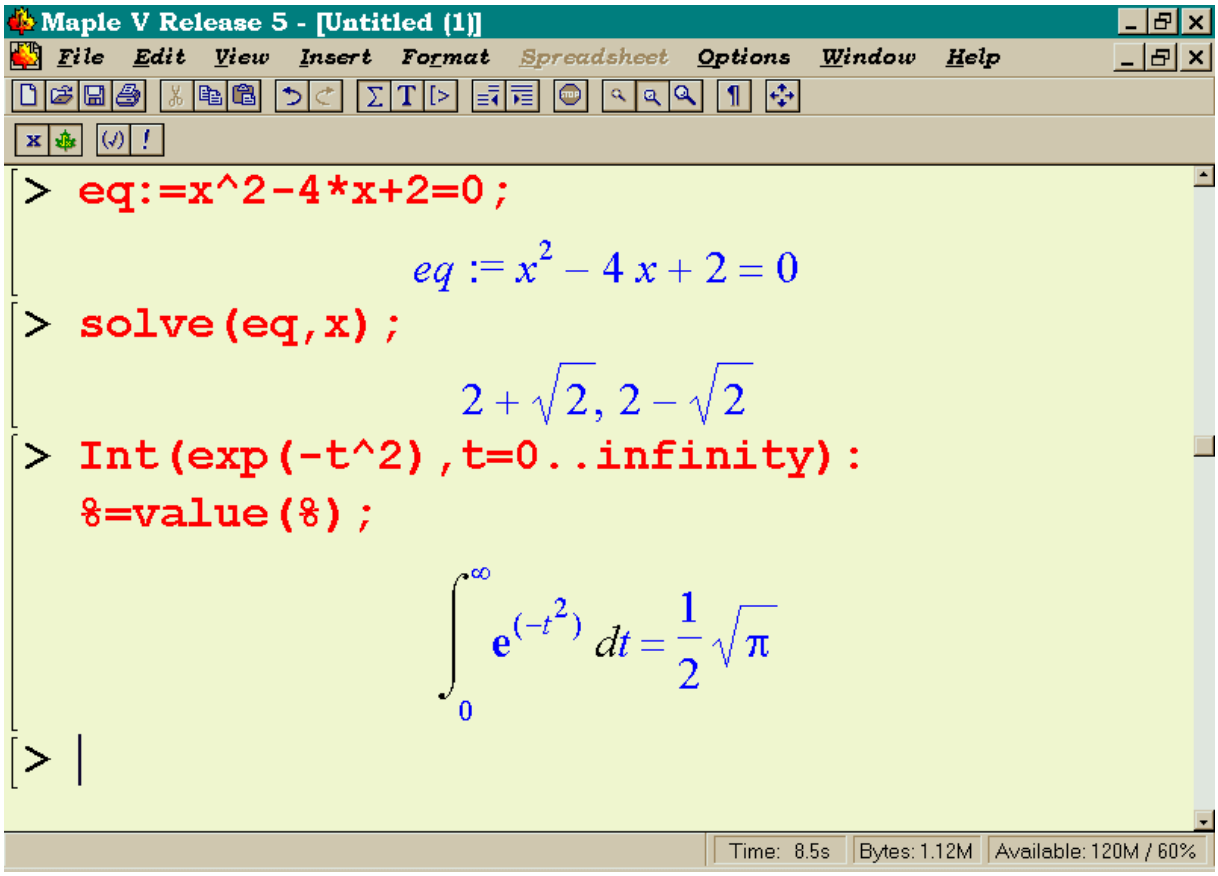
Autres actions

[Ctrl] + [Z] annule la dernière action.
[Ctrl] + [Y] réalise à nouveau une action annulée.
[Ctrl] + [N] nouveau plan de travail.
[Ctrl] + [O] ouvrir un plan de travail sauvegardé.
[Ctrl] + [S] sauvegarde un plan de travail.
[Ctrl] + [P] imprime un plan de travail.
[Alt] + [F4] Fermeture du programme

2°) Raccourcis particuliers à Maple

[Ctrl] + [K] insère une ligne de commande avant le point d'insertion.
[Ctrl] + [J] insère une ligne de commande après le point d'insertion.
[Ctrl] + [Suppr] : supprime la ligne de commande en cours.
[Ctrl] + [C] en l'absence de sélection, copie l'intégralité de la ligne de commande (ne marche pas sur tous les systèmes).

VII. Ecran Maple



 Nouveau / Ouvrir / Sauvegarder / Imprimer  Couper / Copier / Coller

 Annuler / Répéter la dernière action  Interrompre un calcul  Réglage du zoom