Chapitre 1: Calculer avec Maple

Dans Maple Les opérations +, * et d'élévation à une puissance entières et rationnelles sont effectuées automatiquement. Le résultat est aussi automatiquement réduit à sa plus simple expression.

Soit les expressions $\frac{7}{5}+\frac{3}{2}-\frac{8}{7}, \left(\frac{5}{8}\right)^6$ *et* $\frac{4(18)}{5(29)}$

```
> 7/5+3/2-8/7;
>(5/8)^6;
>4/5*18/29;
```

Les expressions arithmétiques sont simplifiées dans les rationnels **Q** que si elles ne contiennent aucun nombre décimal.

```
> 298*(3^14+7^38)/(1-4^27);
```

Si au moins un des nombres est un nombre décimal, le résultat sera automatiquement un nombre décimal.

```
> 298*(3.^14+7^38)/(1-4^27);
```

Maple opère symboliquement la simplification en accord avec le mécanisme de la simplification automatique.

```
> sqrt(3)+2*sin(Pi/3)+cos(Pi/3);
```

Pour obtenir *une approximation décimale d'une expression exacte* La macro-commande **evalf**. Si on ne précise pas le nombre de chiffres désirés, le résultat sera donné avec le nombre de chiffres spécifié par la variable d'environnement **Digits**. La valeur par défaut de cette variable est 10.

Obtenons une approximation du nombre $298(3^{14}+7^{38})/(1-4^{27})$.

```
> evalf(298*(3^14+7^38)/(1-4^27));
```

Obtenons une autre approximation mais en précisant cette fois que l'on désire 50 chiffres. Il suffit de préciser la valeur 50 en option dans la macro-commande **evalf**. Cet ajout ne sert qu'à préciser ponctuellement le nombre de chiffres désirés sans pour autant modifier la valeur de la variable d'environnement **Digits**.

```
> evalf(298*(3^14+7^38)/(1-4^27),50);
```

Essayer l'expression

```
> evalf(298*(3.^14+7^38)/(1-4^27),50);
```

?

Mécanisme de la simplification automatique

Les nombres rationnels et les radicaux sont réduits automatiquement à leur plus simple expression. Soit les expressions $\frac{123456}{987654}$ et $\sqrt{63}$.

```
> 123456/987654; > sqrt(63);
```

Dans les expressions rationnelles, les facteurs syntaxiquement identiques au numérateur et au dénominateur sont automatiquement simplifiés, et ce sans évaluation (à la condition que l'expression bâtie ne soit pas la règle de calcul des

images d'une fonction).

Soit les expressions $\frac{2x}{x}$ et $\frac{(x+1)^2 x^3}{x(x+1)^3}$.

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

La réduction des termes semblables et les lois des exposants sont automatiquement appliqués

Soit les expressions $a-b+a$ et $(x+y)^2\sqrt{x+y}$.

```
> a-b+a;
> (x+y)^2*sqrt(x+y);
> (x^3)^2;
```

La distributivité de la multiplication sur l'addition est automatique. Soit l'expression $\frac{3(a+b)}{7}$.

```
> 3/7*(a+b);
```

fonctions trigonométriques

Le signe moins devant l'indéterminée dans la valeur absolue, dans les fonctions trigonométriques sinus et cosinus et dans les fonctions trigonométriques réciproques est automatiquement simplifié. Soit les expressions $|-x|$, $\sin(-x)$, $\cos(-x)$, $\cot(-x)$, $\arcsin(-x)$ et $\arccos(-x)$.

```
> abs(-x);>sin(-x);>cos(-x);>cot(-x);>arcsin(-x);
>arccos(-x);
```

Les identités trigonométriques des angles complémentaires et des angles supplémentaires et autres angles sont utilisées pour réduire automatiquement les expressions trigonométriques. Soit les expressions

$\tan\left(x+\frac{\pi}{2}\right)$, $\sin\left(x+\frac{\pi}{2}\right)$, $\cos(x+\pi)$, $\sin\left(-x-\frac{\pi}{4}\right)$ ***et*** $\sin\left(-x+\frac{\pi}{5}\right)$.

```
> tan(x+Pi/2);>sin(x+Pi/2);>cos(x+Pi);>sin(-x-Pi/4);
>sin(-x+Pi/5);
```

L'argument des fonctions trigonométriques doivent être ***précisés en radians***.

```
> sin(13*Pi/4);
```

Si l'angle est en degrés, nous devons alors le convertir en radians en utilisant la macro-commande **convert**.

```
> cos(225*degrees);
> cos(convert(225*degrees,radians));
> 225*degrees = convert(225*degrees,radians);
```

Le logarithmes

Tout calcul de logarithmes dans une base autre que la base ***e*** est automatiquement converti dans la base ***e***

```
> log[a](12);
```

De plus, l'expression $e^{\ln(x)}$ est simplifiée automatiquement par x, mais l'expression $\ln(e^x)$ ne l'est pas. La définition de la fonction logarithme dans les nombres complexes ne permet pas cette simplification automatique.

```
> exp(ln(x));> ln(exp(x));
```

Maple comporte une table d'affectation système qui lui permet de simplifier automatiquement plusieurs expressions trigonométriques, logarithmiques et d'autres expressions numériques remarquables

Soit les expressions numériques $\cos\left(\frac{\pi}{6}\right)$, $\tan\left(\frac{17\,\pi}{3}\right)$ et $\ln(1)$.

```
> cos(Pi/6);
>tan(17*Pi/3);
ln(1);
```

Maple simplifie donc automatiquement sous forme de radicaux les multiples entiers des arcs remarquables:

$$0, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}, \frac{\pi}{2}, \pi$$

Pour exprimer le résultat avec des radicaux lorsque π est divisé par un diviseur de 120 ou de 24, il faudra faire une simplification sur demande en utilisant la macro-commande `convert` (voir ?convert,radical) .

```
> sin(5*Pi/12);
>cos(7/60*Pi);
> convert(sin(5*Pi/12),radical);
>convert(cos(7/60*Pi),radical);
```

La constante mathématique e est saisi avec la macro-commande **exp**.

```
> exp(1);
> evalf(%,20);
> evalf(%,30);
```

?

Les nombres complexes

Pour définir un nombre complexe, on utilise le nombre imaginaire i, que maple représente par I

```
> z := 6+5*I:
```

Pour obtenir ses parties réelles et imaginaires

> **Re(z);Im(z);**
Le conjugué, le module et l'argument d'un nombre complexe z
>**conjugate(z);abs(z);argument(z);**

Pour forcer Maple à écrire un nombre complexe sous forme cartésienne

> z *(1+3*I) ;

> evalc(%) ;

Pour définir un nombre complexe sous forme polaire

> polar(2,Pi/3) ;

> evalc(%) ;

Pour passer de l'écriture cartésienne à la forme polaire

> polar(1+sqrt(3)*I) ;

ou

> **convert(1+sqrt(3)*I,polar);**

L'opérateur d'affectation ou d'assignation

L'opérateur d'assignation « := » permet de donner un nom à des objets créés avec Maple. En effet, cette capacité de nommer les objets permet des développements qui, autrement, serait laborieux et parfois même impossible. Mais cela rend nécessaire l'obligation de distinguer variable mathématique et variable informatique.

Variable mathématique et variable informatique

Au cours d'une session Maple, il n'y a aucune déclaration à faire lorsqu'on utilise une variable mathématique. L'évaluateur la prend **immédiatement** en charge.

> **x; # la variable x est libre (x est une variable mathématique)**
> **x^2;**
> **sin(theta+2*k*Pi);**

L'opérateur d'affectation ou d'assignation est le deux-points égal « := »

> **x:=-5; # La variable x n'est plus libre (x est maintenant une variable informatique)**
> **x^2;**

Le résultat n'est pas x^2 mais 25 car x, en tant que variable informatique, pointe vers la valeur 5.

Lorsqu'une variable est assignée, l'assignation demeure active tout au long de la séance de travail tant et aussi longtemps qu'aucune autre assignation ne viendra la modifier.

Au cours d'une session Maple, on peut rendre à nouveau une variable mathématique à l'aide des *apostrophes droits* (***quotes***).

```
> x:='x';
```

Voyons voir.

```
> x^2;
```

Les apostrophes droits commandent à l'évaluateur de limiter l'évaluation au nom de la variable. Par contre, cette limitation ne limite en aucun temps le mécanisme de la simplification automatique en excluant la simplification des entrées de la table d'affectation système.

```
> sqrt(4^2+a^2+b+b+cos(0));
>'sqrt(4^2+a^2+b+b+cos(0))';
```

Un des effets de la macro-commande **restart** est de désassigner toutes les assignations courantes.

Remarque: Avant d'entreprendre tout autre développement dans une autre section, il est recommandé de rendre libres les variables qui n'ont plus de raison d'être assignées.

```
> x:='x':
>y:='y':
```

Pour désassigner certaines variables seulement sans modifier l'état courant de la session de travail (variables systèmes et macro-commandes spécifiées avec `with`), c'est mieux le faire comme suit à l'aides des apostrophes droits:

```
> x:='x':
>toto:='toto':
```

Il est possible aussi d'utiliser la macro-commande **unassign** pour désassigner simultanément plusieurs variables.

```
> unassign('x,toto,a,b,c');
```

La commande *radnormal*

La macro-commande **radnormal** est la macro-commande dédiée à la simplification de nombres comportant des radicaux. Par exemple:

```
> Formule:= (1-1/5*5^(1/2))*(2/(5^(1/2)-1))^12/(5^(1/2)-
1)+(-1-1/5*5^(1/2))*(-2/(5^(1/2)+1))^12/(5^(1/2)+1);
> radnormal(Formule);
```

Il est même possible d'effectuer la rationalisation du dénominateur:

```
> Formule:=(2+sqrt(6))/(sqrt(3)+sqrt(2));
> radnormal(Formule);
```

L'option `rationalized` permettera la rationalisation du dénominateur.

```
> radnormal(Formule,rationalized);
> (2+sqrt(6))/(sqrt(3)+sqrt(2));
> radnormal(%);
> radnormal(%%,rationalized);
```

La simplification sur demande

Par simplification en mathématique, on entend habituellement réduction d'une expression. La macro-commande **simplify** est, en générale, la première macro-

commande à laquelle on pense faire usage lorsqu'on veut simplifier une expression mais elle ne produit pas nécessairement le résultat attendu.

```
> Formule:=(2+sqrt(6))/(sqrt(3)+sqrt(2));
> simplify(Formule);
```

Par simplification d'une expression, il faudrait plutôt entendre transformation de l'expression et cela, habituellement, vers une écriture plus appropriée à un contexte donné.

```
> Formule:=1-(sin^2)(x);
> simplify(Formule);
```

Mais c'est pas toujours le cas avec *simplify*.

```
> Formule:=sin(2*x)/cos(2*x);
> simplify(Formule);
```

Maple, de manière automatique ou sur demande n'estime pas que $\tan(2x)$ est une écriture plus simple. Utilisons alors la macro-commande **convert.**

```
> convert(Formule,tan);
```

La macro-commande **convert** est donc une macro-commande de simplification. Mais, très souvent, `simplify` donne de bonne réduction.

```
> Formule:=((sin^3)(x)+(cos^3)(x))/(sin(x)+cos(x));
> simplify(Formule);
```

Sans élaborer plus avant sur la notion de simplification sur demande, mentionnons que les principales macro-commandes disponibles pour réaliser des simplifications sur demande sont

radnormal, **normal**, **simplify**,
expand, **factor**, **combine**,
collect, **convert**, **rationalize**,
sort, **subs**, **algsubs**.

les macro-commandes evalf et eval

La macro-commande **evalf** est utile pour obtenir une approximation décimale d'une expression exacte tandis que la macro-commande **eval** sert à évaluer une expression dont le résultat peut être exacte.

```
> Formule:=sqrt(exp(x));
> evalf(Formule,x=Pi);
```

message d'erreur ?

En effet, il faut d'abord évaluer l'expression puis obtenir après seulement une approximation de cette évaluation.

```
> eval(Formule,x=Pi);
> evalf(eval(Formule,x=Pi));
```

La dernière requête a permis d'obtenir une approximation de la valeur exacte de `eval(Formule,x=Pi)`, c'est-à-dire de $\sqrt{e^{\pi}}$.

De plus, sur un autre plan, il est utile d'utiliser la macro-commande **`eval`** pour bien documenter la zone des résultats.

```
> eval(Formule,x=Pi)=evalf(eval(Formule,x=Pi));
```

Les variables

Il ne faut pas perdre de vue, au moment de la création ou de l'utilisation d'une formule, si les variables utilisées sont libres (variables mathématiques) ou si elles sont assignées (variables informatiques) car cela peut mener à des erreurs «logiques».

```
> x:=2;
> eval(x/(x+1),[x=100]); # Ici les crochets sont facultatifs
```

Compte tenu de l'état du système au moment de la requête d'évaluation, `Maple` a "évalué" la constante 2/3. Maple n'a pas pu évaluer la formule $x/(x+1)$ avec $x=100$ comme on aurait voulu qu'il le fasse puisque, par l'assignation $x:=2$, la variable x était informatique. Ainsi, la variable x pointant vers la valeur 2, la formule $x/(x+1)$ pointe alors vers le nombre 2/3 au moment de l'évaluation avec la macro-commande `eval`.

Pour effectuer l'évaluation attendue, il faut rendre d'abord la variable x mathématique, c'est-à-dire libre.

```
> x:='x';
```

Ensuite, l'évaluation pourra être faite correctement.

```
> eval(x/(x+1),[x=100]);
```

Variable d'environnement *Digits* et chiffres significatifs

La variable **Digits** est une variable d'environnement qui impose à Maple le nombre de chiffres qui sera utilisé dans les approximations demandées. Sa valeur par défaut est 10 mais l'usager peut, en tout temps, lui assigner le nombre voulu de chiffres.

Une remarque s'impose immédiatement: **il ne faut pas confondre nombres de chiffres utilisés dans les calculs et nombre de chiffres significatifs du résultat**.

Une fonction en Maple

L'opérateur fonctionnelle flèche « -> » est à utiliser pour la création de fonctions. L'opérateur flèche est utilisé pour transposer en Maple la notion mathématique de règle de correspondance fonctionnelle.

```
> x->2*x-3;
```

Une procédure est en quelque sorte la transposition informatique du concept mathématique de fonction. Obtenons l'image du nombre 5 par la formule de calcul des images $2x-3$;

```
> (x->2*x-3)(5);
```

Mais, comme en mathématique, il est commode de donner un nom à une fonction. Donnons-lui le nom f. Pour ce faire, assignons donc la fonction $x \to 2x-3$ à la variable f.

```
> f:=x->2*x-3;
```

Ainsi, $f(x)$ désignera l'image de x par la fonction f.

```
> f(x);
> f(6);
> f(t);
>f(x+h);
> f:='f':  # Rendons libre la variable f
```

Résolution d'équations et notation indicielle

La résolution d'une équation est réalisée avec la macro-commande **solve**.

```
> Eq_1:=(x/(x-1))^2=6*(x/(x-1))+7;
```

Avec la macro-commande `solve`, il y a deux syntaxes permise: avec ou sans l'utilisation des accolades autour de la variable de résolution.

```
> solve(Eq_1,{x});
> solve(Eq_1,x);
```

Un autre exemple.

```
> f:=x->2*x^2+5*x+6;
> Racines:=solve(f(x)=0,x);
```

En donnant un nom au résultat, il est plus commode ensuite de pointer vers l'un ou l'autre des éléments du résultat. En effet, Racines étant une **séquence**, il suffit d'utiliser la notation indicielle Racines[1] et Racines[2] pour pointer respectivement vers les premier et le second élément de Racines.

```
> Racines;
> Racines[1];
> Racines[2];
```

Vérifions si ce sont bien deux zéros de la fonction f.

```
> Verification1:=f(Racines[1]);
>Verification2:=f(Racines[2]);
```

Remarquons qu'il y a tout de même eu simplification automatique. Nous pouvons, encore, opérer une simplification sur demande avec **radnormal**.

```
> radnormal(Verification1);
>radnormal(Verification2);
```

Résolution d'inéquations

```
> Inéquation:=x^2+6*x+5<0;
> E_S:=solve(Inéquation,x);
> E_S:=solve(Inéquation,{x});
```

Observer encore ici la différence dans la présentation du résultat de la macro-commande `solve` selon que les accolades sont utilisées ou non.

L'ensemble solution obtenu de l'évaluateur est un objet Maple qu'il nous faut interpréter car c'est un objet Maple de type ensemble (**set**) qui contient deux objets de type `≤` . L'ensemble-solution n'est donc pas, pour Maple, un ensemble de nombre réels.

```
> member(-3,E_S);
> member(-5<x,E_S);
```

La macro-commande **solve** est aussi utile pour résoudre un système d'équations ou un système d'inéquations.

Résolvons le système d'équations suivant:

$$\begin{aligned} -2x + y + 2z + w &= 12 \\ y - 2z - 3w &= -20 \\ 2x + y + 3z + 2w &= 12 \\ -x - y - 3z + w &= 28 \end{aligned}$$

```
>eq1:=-2*x + y + 2*z + w =12;
>eq2:= y - 2*z  - 3*w = -20;
>eq3:= 2*x + y + 3*z + 2*w  = 12;
>eq4:= -x - y - 3*z + w  = 28;
> solve({q1,q2,q3,q4},{x,y,z,w});
```

Remarque: Attention à l'ordre alphabétique des variables.

Fonctions rationnelles

Il est **possible** de créer une fonction rationnelle avec des facteurs communs explicitement au numérateur et au dénominateur. Le mécanisme de la simplification automatique le permet.

```
> f:=x->(x+2)*(x^2+2*x+3)/(x+2);
> 'f'(x)=f(x);
```

Par contre,

```
> f(-2);
```

Bien qu'il y a affichage de $f(x) = x^2 + 2x + 3$, Maple a retenu que

f: $x \rightarrow \dfrac{(x+2)(x^2+2x+3)}{x+2}$.

Macro-commande *piecewise*

La macro-commande **piecewise** est utile pour créer des formules de calculs par morceaux.

```
> Formule:=piecewise(x<2,x^3+1,x>3 and x<5,4,x>5,1/x);
> eval(Formule,x=10);
> eval(Formule,x=0);
> eval(Formule,x=3);
```

?

Maple donne automatiquement la valeur 0 associée aux valeurs de x qui ne sont pas conditionnées (pas dans le domaine). Ce qui n'est pas pratique.

```
> simplify(Formule);
```

On peut préciser, en option, dans la macro-commande `piecewie`, une "valeur" qui sera pris en compte pour toutes les valeurs de x qui ne sont pas conditionnées. Si on veut que l'expression ne soit pas définie, il suffit de précisiser comme dernier argument dans la macro-commande `piecewise`, une variable libre, la variable "non_définie" par exemple.

```
> Formule:=piecewise(x<2,x^3+1,x>3 and
x<5,4,x>5,1/x,non_définie);
> simplify(Formule);
```

évaluons maintenant Formule avec $x = 3$.

```
> eval(Formule,x=3);
```

Que faut-il comprendre dans les deux évaluations suivantes.

```
> eval(Formule,x=infinity);
> eval(Formule,x=-infinity);
```

?

Retour sur le mécanisme de la simplification automatique

Dans les nombres réels, l'expression $\sqrt{x^2} = |x|$. Qu'en est-il de cette simplication avec Maple.

```
> Expression:=sqrt(x^2);
> simplify(Expression);
```

Ce qui nous indique qu'en général, nous n'avons pas $\sqrt{x^2} = x$ ni même $\sqrt{x^2} = |x|$.

Afin d'obtenir la simplification attendue, soit $|x|$, il faut spécifier à Maple de faire la simplification demandée en tenant compte que la variable *x* représente un nombre réel.

```
> simplify(Expression, assume=real);
```

Montrons, en effet, que dans **C**, $\sqrt{x^2} \neq x$ et $\sqrt{x^2} \neq |x|$.

```
> x:=-1-I;
> sqrt(x^2)<>x;
> sqrt(x^2)<>abs(x);
> x:='x': # Rendons x libre pour le prochain exemple
```

Présentons une autre situation de simplification automatique inattendue.

Dans **R**, on a, bien sûr, $\mathbf{e}^{\ln(x)} = x$ et $\ln(\mathbf{e}^x) = x$. Mais, qu'en est-il avec Maple. Ne pas oublier que pour Maple, le référentiel est l'ensemble des nombres complexes.

```
> exp(ln(x));
> ln(exp(x));
```

Avec Maple, $\ln(\mathbf{e}^x)$ n'a pas été automatiquement simplifiée par *x*, parce que ce n'est pas toujours vraie. C'est-à-dire que l'égalité $\ln(\mathbf{e}^x)$ = x n'est pas une identité dans **C**.

```
> ln(exp(-3-5*I));
> evalc(%);
```

Ce qui montre que $\ln(\mathbf{e}^{(-3-5I)}) \neq -3 - 5I$.

Pour chercher la solution de $\ln(\mathbf{e}^x) = x$ dans **R** seulement.

```
> simplify(ln(exp(x)),assume=real);
```