Physique Numérique (SMP6)

TP1 : PREMIERS PAS EN MAPLE

Qu'est-ce que Maple ?
Maple est un logiciel de calcul formel : il est capable de manipuler des nombres, ou des objets mathématiques plus compliqués (fonctions, matrices, équations, tableaux, procédures...), le tout de façon "formelle", i.e. sans avoir à évaluer la valeur de ces objets, en manipulant des expressions symboliques ne contenant pas que des paramètres numériques.
Par exemple, Maple connait le nombre Pi :

```
> Pi;
```

$$\pi \quad \textbf{(1)}$$

Maple peut effectuer des calculs exacts avec Pi sans en évaluer la valeur numérique :

```
> cos(Pi)^2 + sin(Pi)^2;
```

$$1 \quad \textbf{(2)}$$

Cependant, il est possible d'évaluer Pi en tant que réel (avec une erreur numérique), si nécessaire :

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

$$3.141592654 \quad \textbf{(3)}$$

Le but des TPs : Il s'agit d'apprendre à utiliser Maple ainsi que les rudiments de programmation, dans des cas d'utilisations concrètes.

Présentation de l'interface :

- A l'ouverture de Maple, la feuille blanche est appelée feuille de calcul (*worksheet*). C'est là où vous entrez vos commandes, où Maple les exécute, et donne les résultats correspondants. Elle commence par une invite (*prompt*, symbole >) : cela signifie que Maple est prêt à recevoir les commandes de l'utilisateur.
- Si on tape une commande comme 1+1 et qu'on valide avec la touche Entrée :

```
> 1+1
Warning, inserted missing semicolon at end of statement
```

(4)

...on reçoit un message d'erreur.

REGLE N°1 : une commande doit toujours être conclue par un caractère terminateur, soit un point-virgule (;), soit deux-points (:).

```
> 1+1;
```

$$2 \quad \textbf{(5)}$$

Si on utilise deux-points à la place de point-virgule, Maple effectue le calcul sans afficher le résultat ; ceci est en particulier utile lorsqu'on effectue un calcul dont le résultat est long et/ou qu'il n'est pas nécessaire de l'afficher :

```
> 80!; # je veux voir le résultat
```

(6)

715694570462638022948115337231865321655846573423657525771094450582270392554801\
14884266894486728081408000000000000000000 **(6)**

```
> 80!: # je ne veux pas voir le résultat
```

On peut entrer plusieurs commandes sur la même ligne :

```
> cos(0); sqrt(27);
```

$$1$$

$$3\sqrt{3} \quad \textbf{(7)}$$

- Pour enregistrer sa feuille de travail, faire : "Fichier", "Enregistrer sous". Le fichier a une extension .mw.
- Pour ouvrir une feuille enregistrée, on utilise le menu "Fichier", "Ouvrir". Si on souhaite travailler à nouveau dessus, il est nécessaire de ré-exécuter toutes les lignes de commande, soit en les validant une à une avec la touche "Entrée", soit en utilisant le menu "Edition", "Exécuter", "Feuille de travail".

Utilisation de l'aide

Il faut comprendre que Maple est un logiciel très riche. Il est hors de question de connaître toutes les commandes et leurs syntaxes (d'autant plus que la syntaxe peut changer d'une version de Maple à l'autre).

REGLE N°2 : Face à une commande inconnue, on utilisera de manière systématique l'aide de Maple.

- Si on veut des informations sur une commande précise, il suffit de taper un ? suivi immédiatement du nom de la commande (Exception : il n'y a pas besoin de ; terminateur ici) :

```
> ?isprime
```

- Si on ne connait pas le nom de la commande, on utilise "Aide", "Aide Maple".

Affectation des variables

Il est pratique de donner des noms à des résultats antérieurs, pour pouvoir les réutiliser par la suite : c'est l'*affectation*. En voici un exemple :

```
> produit:= 7!;
```

$$produit := 5040 \quad \textbf{(8)}$$

A gauche du signe :=, on entre le nom de la variable et à droite la valeur affectée. Comprenons bien ce qu'il s'est passé ici : nous pouvons voir les choses de la manière suivante : Maple a réservé un emplacement dans la mémoire de l'ordinateur, qui contient la valeur 5040 dont le nom est "produit". On peut aussi dire que la variable "produit" contient la valeur 5040. On peut vérifier l'affectation par :

```
> produit;
```

$$5040 \quad \textbf{(9)}$$

on peut l'utiliser pour faire de nouveaux calculs :

```
> produit/7;
```

$$720 \quad \textbf{(10)}$$

Notamment, il est important de bien comprendre la commande suivante :

```
> produit:=produit+1;
```

$$produit := 5041 \quad \textbf{(11)}$$

A droite de :=, "produit+1" est la valeur contenue dans la variable "produit" augmentée de 1, c'est à dire 5041. Autrement dit, on a remplacé dans la variable "produit" l'ancienne valeur 5040, par une nouvelle

valeur 5041.

• Le nom de la variable ne doit pas comporter de signes de ponctuation, d'espace, ni de caractères spéciaux (par ex. +,*,&,%,@). On peut utiliser des majuscules ou des minuscules (Attention ! Maple différencie les deux !). Attention : certains noms sont réservés à des fonctions déjà définies par Maple :

```
> D:=2;
```

```
Error, attempting to assign to `D` which is protected
```

La procédure d'affectation est très générale : on peut également nommer des nombres rationnels, des complexes, des fonctions, des matrices...

• Pour réinitialiser (désaffecter) la variable produit et faire en sorte qu'elle ne contienne plus la valeur 5041, on effectue l'une ou l'autre des commandes suivantes :

```
> produit:='produit';
```

$$produit := produit \quad (12)$$

```
> unassign('produit');
```

On vérifie que la variable produit est bien désaffectée :

```
> produit;
```

$$produit \quad (13)$$

Si on veut réinitialiser toutes les variables, on utilise la commande restart.

REGLE N°3 : avant CHAQUE exercice, au début de CHAQUE feuille de travail, on utilisera la fonction restart, pour être sûr que toutes les variables sont désaffectées.

```
> restart;
```

Ordre des commandes

Il est très important de comprendre que le comportement de Maple dépend de l'ordre chronologique d'évaluation des commandes et non de l'ordre d'apparition sur la feuille de calcul.
Dans une feuille de calcul, rien ne vous empêche de modifier une commande entrée précédemment : pour cela, il suffit de remonter à la ligne en question, de modifier la commande et de valider avec "Entrée". Cependant, attention à cette manipulation ! On méditera sur l'exemple suivant :

```
> a:=3;
```

$$a := 3 \quad (14)$$

```
> b:=a/2;
```

$$b := \frac{3}{2} \quad (15)$$

Si en remontant, je décide de modifier a:=3 en a:=2, mais que j'oublie de valider la ligne suivante, b contiendra toujours la valeur 3/2 et ne sera plus égal à a/2 !

REGLE N°4 : Si on modifie une ligne précédente, il faut ré-exécuter toutes les lignes suivantes.

Calculs sur les nombres entiers

Maple fait automatiquement des calculs exacts sur de très grands nombres entiers. Les opérations usuelles sont +,-,*.

Dans Maple, les differents objets ont un type. On peut demander le type d'un objet par la commande whattype. Par exemple, le type d'un entier est *integer*.

```
> whattype(6!);
```

integer **(16)**

Calculs sur les nombres réels

La première façon d'approcher un nombre réel grâce à Maple est d'utiliser le point décimal (.) (c'est l'équivalent anglo-saxon de notre virgule décimale). Maple calcule alors des valeurs approchées, avec un nombre de chiffres significatifs fixé (par défaut 10 ; cf la commande *Digits*).

```
> 300/45;
```

$$\frac{20}{3} \quad \textbf{(17)}$$

Pour Maple, le nombre précédent n'est pas un nombre réel, c'est un nombre rationnel, qu'il a spontanément simplifié. Il faut forcer Maple à effectuer un calcul réel (approché), en faisant :

```
> 300./45;
```

6.666666667 **(18)**

ou bien :

```
> evalf(300/45);
```

6.666666667 **(19)**

Le type d'un réel est *float* (comme nombre *flottant*).

```
> whattype(300/45); whattype(evalf(300/45));
```

fraction
float **(20)**

On peut chosir le nombre de chiffres significatifs (Attention : Majuscule pour Pi !)

```
> evalf(Pi, 20);
```

3.1415926535897932385 **(21)**

Voici quelques fonctions connues par Maple : regardez l'aide pour plus de précisions :
exp
ln ou log
sqrt
sin, cos, tan
abs
trunc, floor, ceil
max, min
erf

Calculs sur les nombres complexes
Pour définir un nombre complexe, on utilise le nombre imaginaire i, que Maple représente par I (Attention, i majuscule !)

```
> z:=3+4*I;
```

$z := 3 + 4\,I$ **(22)**

Pour obtenir les parties réelles et imaginaires :

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

3
4 **(23)**

```
> z*(1+sqrt(2)*I);
```

$(3 + 4\,I)\left(1 + I\sqrt{2}\right)$ **(24)**

Pour forcer Maple à écrire le nombre sous forme cartésienne (Re + i Im), on utilise *evalc* :

```
> evalc(");
```

$$3 - 4\sqrt{2} + I\left(4 + 3\sqrt{2}\right) \quad (25)$$

Les commandes pour obtenir le nombre complexe conjugué, le module et l'argument sont : *conjugate, abs, argument*. L'argument (donné par Maple) est dans]-Pi, Pi].

```
> conjugate(z); argument(z); abs(z);
```

$$3 - 4\,I$$

$$\arctan\left(\frac{4}{3}\right)$$

$$5 \quad (26)$$

Pour définir un nombre complexe sous forme trigonométrique r e^itheta, on utilise la commande *polar* :

```
> polar(3, Pi/6);
```

$$\mathrm{polar}\left(3, \frac{1}{6}\,\pi\right) \quad (27)$$

```
> evalc(%);
```

$$\frac{3}{2}\sqrt{3} + \frac{3}{2}\,I \quad (28)$$

Pour passer de l'écriture cartésienne à l'écriture en polaire, on utilise encore *polar* mais la syntaxe est différente (pour la version Maple V il ne faut pas oublier de déclarer la bibliothèque readlib(polar) :

```
> polar(1+I);
```

$$\mathrm{polar}\left(\sqrt{2}, \frac{1}{4}\,\pi\right) \quad (29)$$

Simplification d'expressions

Lorsqu'un résultat obtenu par un calcul algébrique n'a pas la forme voulue, certaines fonctions de Maple permettent d'y remédier. La plus générale est *simplify*.

```
> 4^(1/2)+4; simplify(%);
```

$$\sqrt{4} + 4$$

$$6 \quad (30)$$

```
> (sin(x))^4 -(cos(x))^4; simplify(%);# (% est remplacée par " dans Maple V)
```

$$\sin(x)^4 - \cos(x)^4$$

$$1 - 2\cos(x)^2 \quad (31)$$

Un exemple important :

```
> y:= sqrt(x^2);
```

$$y := \sqrt{x^2} \quad (32)$$

```
> simplify(y);
```

$$\mathrm{csgn}(x)\,x \quad (33)$$

Quelle est la fonction *csgn* ? Le résultat est-il correct ? Par défaut, Maple ne sait rien de la variable non affectée x et la considère comme un nombre complexe. Si on suppose (assume en anglais) que x est positif, on peut encore simplifier l'expression :

```
> simplify(y)assuming x>0;
```

$$x \quad (34)$$

```
> simplify(y, assume=positive);
```

$$x \tag{35}$$

La supposition est temporaire (le temps que la commande soit effectuée). Si on souhaite qu'elle soit permanente, on utilise assume avec une syntaxe différente :

```
> assume(x, positive); simplify(y);
```

$$x\sim \tag{36}$$

Le tilde est là pour rappeler qu'une hypothèse a été faite sur x. La commande *about* affiche les hypothèses faites sur x :

```
> about(x);
Originally x, renamed x~:
  is assumed to be: RealRange(Open(0),infinity)
```

D'autres commandes existent pour simplifier les expressions :
- *factor* : la commande *factor* factorise les expressions polynomiales :

```
> restart;factor(x^3+x^2);
```

$$x^2\,(x+1) \tag{37}$$

```
> factor(x^2+1, complex);
```

$$(x+1.000000000\,\mathrm{I})\,(x-1.\,\mathrm{I}) \tag{38}$$

```
> factor(y^4-2,sqrt(2));
```

$$-\left(-y^2+\sqrt{2}\right)\left(y^2+\sqrt{2}\right) \tag{39}$$

- *expand* : la commande *expand* développe les expressions :

```
> expand((x+y)*(z+t));
```

$$x\,z+x\,t+y\,z+y\,t \tag{40}$$

```
> expand(cos(3*t));
```

$$4\cos(t)^3-3\cos(t) \tag{41}$$

- *combine* : permet de combiner une expression (voir l'aide pour les options éventuelles) :

```
> A:= cos(p)*cos(q); combine(A);
```

$$A := \cos(p)\,\cos(q)$$

$$\frac{1}{2}\cos(p-q)+\frac{1}{2}\cos(p+q) \tag{42}$$

- *subs* : permet de substituer une expression dans une autre : exemples :

```
> subs(3=6, 3*x+1);
```

$$6\,x+1 \tag{43}$$

```
> B:= 1/(1+tan(a)^2); B:=subs(tan(a)=cos(a)/sin(a), B); B:=combine
  (B);
```

$$B := \frac{1}{1+\tan(a)^2}$$

$$B := \frac{1}{1+\dfrac{\cos(a)^2}{\sin(a)^2}}$$

$$B := \frac{1}{2}-\frac{1}{2}\cos(2\,a) \tag{44}$$