Introduction à Maple

Nicolas Gast - nicolas.gast@ens.fr

19 septembre 2005

1 Introduction

1.1 Plan de l'année

1. Premier contact avec Maple
2. Variables et structures de données
3. Fonctions, utilisation du graphisme
4. Eléments de programmation et boucles itératives et conditionnelles
5. Suites
6. Fonctions et Développement limités
7. Primitives et intégrales
8. Approximation et interpolation

1.2 Que trouve t'on sur un ordinateur ?

Un ordinateur peut se décomposer en deux parties : l'architecture physique (hardware) et l'architecture logicielle (software).

L'architecture physique est en ensemble complexe qui fonctionne grace à un système d'exploitation. Le système d'exploitation est le programme central qui contient les éléments de base nécessaires au bon fonctionnement de l'ordinateur.

Le système d'exploitation alloue les ressources physiques de l'ordinateur (temps processeur, mémoire etc.) aux différents programmes en cours d'exécution. Il fournit aussi des outils aux autres programmes (comme les drivers) afin de leur faciliter l'utilisation des différents périphériques sans avoir à en connaître les détails physiques.

L'utilisateur peut ensuite utiliser l'ordinateur au moyen d'applications, qui permettent de faire communiquer un être humain avec le hardware. C'est à ce niveau là que nous allons jouer.

1.3 La programmation

La programmation est l'ensemble des activités qui permettent l'écriture des programmes informatiques. L'utilisateur est ammené à écrire un “code” informatique, écrit dans un *langage de programmation* qui est ensuite exécuté par l'ordinateur.

On peut distinguer (au moins) 3 niveaux de programmation :

– *Le langage assembleur* : c'est le langage le plus proche du langage machine : les instructions sont du type mettre la valeur a dans la zone mémoire b, incrémenter,. . .. L'assembleur n'a aujourd'hui que très peu d'intéret et n'est pratiquement plus utilisé.

– *Les langages compilés* : dans la plupart des langages de programmation (par exemple c, c++, caml, fortran,. . .), les programmes sont écrits dans un langage lisible par un être humain puis transformé par un *compilateur* en un code qui sera directement éxécuté par la machine.

– *Les langages interprétés* : dans un langage comme celui présent dans Maple, les instructions sont directement interprétées par le programme dans lequel elles sont éxécutée et non transformées en un code plus obscure. Généralement les codes produits de cette façon sont plus lents mais plus rapide à déboguer.

Un algorithme est un ensemble d'instructions (écrites dans un langage courants) permettant de résoudre un problème. Il est indépendant de tout langage de programmation. Avant d'écrire un programme, il est toujours fortement conseillé d'écrire sur un papier l'algorithme que l'ont cherche à implémenter. Un algorithme très connu est l'algorithme d'Euclide qui calcule le PGCD de deux nombres :

```
PGCD (a,b)
  si a < b
     alors retourner PGCD (b,a)
```

```
sinon
   si b = 0
       alors retourner a
   sinon
       soit r le reste de la division euclidienne de a par b
       retourner PGCD (b,r)
   finsi
finsi
```

2 Utilisation de Maple

En Maple, une commande se finit toujours par ; ou :. Si elle se finit par :, Maple n'affiche pas le résultat. Il est possible de regrouper plusieurs commandes sur la même ligne.

```
> 5+5; 4*4; 3-3:
                                  10
                                  16
>
```

La commande % permet de rappeler le dernier résultat (affiché ou non), %% rappelle l'avant dernier résultat.

```
> 5+5;
                                  10
> %*2;
                                  20
>
```

Pour multiplier 2 expressions, il faut toujours mettre un * entre les deux, idem pour la division : /.

```
> 4*4;
                                  16
>
```

Dans Maple comme dans tout langage de programation, il possible d'attribuer une valeur à une variable. Attention, le nom d'une variable commence toujours par une lettre et peut ensuite être suivie de chiffres. ATTENTION : les majuscules ont une importance.

```
> toto23 := 8; ToTo23 := 6;
                              toto23 := 8
                              ToTo23 := 6

> toto23; ToTo23; toto23 * 2;
                                   8
                                   6
                                  16
>
```

Une fois assignée, une variable conserve sa valeur jusqu'à la fin de la session Maple. Il est possible de réinitialier toutes les variables par la commande restart ou une seule avec unassign.

```
> toto := 2: titi := 3:
> toto; unassign('titi'); titi;
                                   2
                                 titi
> restart;
> toto;
                                 toto
```

Afin de rendre plus lisible un programme ou une feuille de calcul, il est impératif d'ajouter un certain nombre de commentaires tout au long des calculs. Un commentaire commence par # et va jusqu'à la fin de la ligne. Les commentaires sont ignorés par Maple.

```
> toto := 2; #on met la valeur 2 à toto
                               toto := 2
>
```

La commande ? permet d'acceder à l'aide de Maple. Pour l'utiliser, taper " ? commande".

```
> ? restart;
restart
clear internal memory

Calling Sequences
     restart
[...]
```

3 Calcul numérique et calcul symbolique

On désigne par calcul numérique une suite d'oppérations arithmétiques mettant en jeu des valeurs numériques (*ie* des nombres).

Il se différencie du calcul symbolique qui autorise l'utilisation de symboles, définis ou non (x, cos, sin, $\int$, ...)

Généralement le calcul numérique donne un résultat *approché*, tandis que le calcul symbolique donne un résultat *exact*. Le calcul symbolique est beaucoup plus puissant que le calcul numérique.

```
> x^2+x^3 - (2*x + x^2);  #ceci est un calcul symbolique
                                   3
                                  x  - 2 x
> x := Pi; cos(3*x/2); # ceci est un calcul numérique
                                     0
>
```

Maple essaie par défaut de rendre un résultat exact et donc pas nécessairement sous forme numérique. Pour le forcer à évaluer les résultats exacts, il y plusieurs méthodes :

- evalf
- marquer un nombre sous forme "flottante" (0.5 au lieu de 1/2)

Le nombre de chiffres après la virgule est défini par la variable Digits. Par défaut elle est remise à 10 par la commande restart ou à chaque nouvelle feuille de calcul.

```
> 1/2 + 2; evalf(%); 0.5 + 2;
                                      5/2
                                  2.500000000
                                      2.5
> evalf(cos(1));
                                 0.5403023059
> Digits := 40: evalf ( cos(1));
                  0.5403023058681397174009366074429766037323
```

Maple dispose de nombreux outils permettant de travailler sur des expressions formelles. Voici par exemple 3 commandes fondamentales :

- expand(expr, expr1, expr2, ..., exprn) : développe les expressions
- simplify(expr) : tende de simplifier la formule.
- facor(expr) : factorise l'expression.

```
> simplify (exp( 3+b*log(a)));
                                          b
                                  exp(3) a
> expand((x+1)*(x+2));
                                  2
                                 x  + 3 x + 2
> factor(5* x^3 - 22*x^2 + 18*x-1);
                                      2
                         (x - 1) (5 x  - 17 x + 1)
```

Maple connait de nombreuses fonctions mathématiques (beaucoup plus que vous, même d'ici 2 ans) ainsi que de nombreuses valeurs. On peut citer par exemple les fonctions trigonométriques ($\cos, \sin, \tan, \arcsin, \arccos, \tanh, \arctan, \ldots$), $\ln, \log[a](b)$, $\exp(x)$. A noter que π s'écrit Pi, que e s'écrit $\exp(1)$. Maple peut aussi calculer sur des complexes, le i complexe s'écrivant I.

```
> cos(Pi);
                                   -1
> arctan(1);
                                   Pi
                                  ----
                                   4
> evalf(exp(10));
                 22026.465794806716516957900645284244366353
> log[10](100);
                                    2
> (1+I)^4;
                                   -4
```

Maple peut aussi résoudre des systèmes d'équations en utilisant la commande solve.

```
> solve(x^3+x^2+x +1= 0, x);
                               -1, I, -I
```

Pour tracer une courbe, il faut utiliser la commande plot (expr, intervalle).

```
> plot(x^3+x+1, x=0..1);

                                [graphe]

> plot(cos(x/(5+x))+3*x-2, x=0..1, y=-1..5);
                                [graphe]
```

4 Exercices

Exercice 1.

- Afficher le résultat de $\cos(\frac{3\pi}{5})$ de façon exacte puis avec 7 chiffres après la virgule
- Calculer $\cos(\frac{x\pi}{2})$ pour $x = 1, 2$ et 3 avec 7 chiffres significatifs

Exercice 2

- Développer $\cos(2x) + \sin(2x)$
- Montrer que

$$\arctan(a) = \arctan(b) + \arctan(\frac{a-b}{1+ab})$$

Exercice 3

Montrer que
- $cos(a)sin(b) = \frac{1}{2}(sin(a+b) - sin(a-b))$
- $cos(a)cos(b) = \frac{1}{2}(cos(a+b) + cos(a-b))$

Exercice 4

Tracer $cos(2) + sin(x/2)$ pour x variant entre -5 et 4.
Tracer $3x^3 - 4x + 10$ pour x variant entre -2 et 1.
- Combien ce polynôme a-t-il de racines réelles ?
- Trouver une approximation de sa (ses) racine(s) a 10^-2 près.

Exercice 5

(Si vous en êtes arrivé là, appelez moi) Implémenter l'algorithme d'Euclide vu plus haut.