Chapitre 1

Introduction au calcul formel et Maple

1. Introduction au calcul formel

1.1 Présentation

On voit souvent un ordinateur comme un calculateur, c'est à dire une machine calculant les résultats d'opérations sur des nombres. Mais un ordinateur peut également manipuler des symboles. Il peut même manipuler des formules, des équations,... sur ces symboles. Ces manipulations sont importantes pour des applications très variées. Le premier intérêt est de pouvoir obtenir des résultats exacts (c'est à dire sans approximation). Le second est d'obtenir des résultats impossibles par un calcul numérique (par exemple la simplification d'une expression ou d'une fraction, l'expression d'une primitive) mais très utiles pour des applications en physique, mécanique,...

1.2 Qu'est-ce que le calcul formel ?

Selon le contexte, l'expression calcul formel - on dit aussi calcul symbolique, plus rarement calcul mathématique assisté par ordinateur - a des sens différents. Nous en distinguons trois. Quand on dit « Delaunay a fait un calcul formel », on veut dire par là un calcul symbolique, par opposition à un calcul purement numérique. Dans la phrase « le calcul formel est en pleine évolution », on désigne la discipline recouvrant les opérations symboliques sur ordinateur. Enfin, quand on parle d'un « système de calcul formel », cela signifie un logiciel permettant de faire des calculs mathématiques exacts, c'est-à-dire à peu près ce que l'on apprend en classe préparatoire aux grandes écoles scientifiques ou dans le premier cycle des universités, ce qui ne l'empêche pas de savoir faire aussi du calcul numérique et des tracés graphiques.
L'intérêt du calcul formel ne se limite pas à la possibilité, certes très spectaculaire, d'obtenir des expressions littérales et des formes closes quasi instantanément pour des calculs fastidieux de dérivées, de primitives, de développements limités et la résolution d'équations algébriques ou d'équations différentielles. La deuxième capacité essentielle d'un logiciel de calcul formel consiste à représenter de manière exacte des objets mathématiques, principalement des nombres, que les logiciels de calcul numérique ne peuvent représenter que de manière approchée, donc avec une certaine erreur qui empêche l'exploitation ultérieure de leurs éventuelles propriétés.
Terminologies employées : calcul formel, calcul symbolique, algèbre computationnelle. Le troisième terme (computer algebra) met bien l'accent sur le domaine mathématique le plus concerné par le calcul formel, l'algèbre.

1.3 Quelques logiciels de calcul formel

Parmi les systèmes de calcul formel existants on peut citer les suivants :
Gap : http ://www.gap-system.org/
Hartmath : http ://www.hartmath.com/
Maple : http ://www.maplesoft.com/
Mathematica : http ://www.wolfram.com/
Mathemagix : http ://www.mathemagix.org/

Maxima : http ://www.maxima.fr.st/
Mupad : http ://www.mupad.de/
Paris-gp : http ://www.parigp-home.de/
Yacas : http ://www.xs4all.nl/ apinkus/yacas.html

2. Introduction à Maple

2.1 Composants du logiciel Maple

Depuis la version 5 de Maple, le logiciel offre un ensemble d'outils qui se déclinent en trois composants principaux : La feuille de travail (worksheet), l'interpréteur de commandes et le langage de programmation.

2.1.1 La feuille de travail

La feuille de travail (*Maple* ***worksheet***) est le composant auquel l'utilisateur est confronté dès qu'il a lancé le logiciel. Son environnement peut prendre différentes formes comme on le verra un peu plus loin. Dans les modes évolués, c'est-à-dire autres que le mode ligne de commande, l'interface présente des menus, des barres d'outils, des menus contextuels, ou encore des fenêtres graphiques permettant à l'utilisateur d'interagir avec le système. Le logiciel n'existe que dans une version anglaise et donc tous les menus sont en anglais. Au fur et à mesure des interventions de l'utilisateur, la feuille de travail, initialement vierge, va afficher les traces des interactions entre l'utilisateur et le logiciel. Dans la rubrique *Save* du menu *File*, figurent différentes manières d'enregistrer ou d'exporter le travail effectué. En particulier, en choisissant de sauvegarder la feuille de travail selon le mode *Maple classic worksheet*, on se ménage la possibilité de retrouver les calculs que l'on vient d'effectuer au cours d'une session ultérieure, par exemple en recourant à la rubrique *Open* du menu *File*.

2.1.2 L'interpréteur

L'interpréteur va déclencher des calculs et afficher des résultats selon les instructions saisies par l'utilisateur. Les instructions font appel à des commandes, c'est-à-dire à des fonctions faisant partie du noyau ou des bibliothèques de Maple. Dans les modes évolués, les résultats mathématiques sont affichés via un *pretty-printer* qui propose un rendu très proche des notations mathématiques usuelles. L'interaction directe avec Maple à travers cet interpréteur consiste à utiliser le logiciel comme une super-calculatrice symbolique et numérique.
Il convient par ailleurs de prendre conscience du fonctionnement de l'interpréteur de Maple. Lorsqu'on commence une session, la mémoire de travail du système ne comporte aucune information. Au fur et à mesure des interactions entre l'utilisateur et le système, la mémoire de travail va s'enrichir des valeurs des variables calculées lors de l'exécution les instructions passées à l'interpréteur. Le lecteur saura bientôt que Maple accepte des variables de types les plus divers. Ainsi, les données passées sous forme de fichiers ou de tableaux, les fonctions programmées par l'utilisateur, … etc. sont naturellement placées dans des variables. Il suffit donc de décrire successivement les différents objets avec lesquels on souhaite travailler pour décrire un problème. Contenus de variables, ces objets vont s'ajouter à l'espace mémoire de travail de Maple qui peut être vu comme une sorte de base de connaissances.

Maple tient compte des informations présentes dans cette base pour répondre aux sollicitations de l'utilisateur. Toutes les variables créées pendant une session sont conservées avec leurs contenus jusqu'à ce qu'une instruction modifie leur valeur ou que l'on décide de procéder à une réinitialisation complète de la mémoire de travail.

2.1.3 Le langage de programmation

Le logiciel met à disposition de l'utilisateur un langage de programmation impératif qui permet d'écrire des fonctions nouvelles élaborées à partir de commandes Maple existantes. Ce langage présente de grandes similitudes dans ses primitives et ses principes de fonctionnement avec des langages comme Pascal ou C. Les mots clés, les structures de contrôle ainsi que les fondements de la programmation avec Maple sont abordés au chapitre suivant.

Arithmétique. Maple connait évidemment les opérations arithmétiques. Il interprete les +, -, / de la même façon que vous et moi.

Fonctions mathématiques. Le nombre de fonctions mathématiques que Maple reconnait est très considérable. En voici un tout petit échantillon : (sin, sqrt, exp, abs, ln, signum)

Calcul. Maple sait dériver, intégrer, prendre la limite, développer une fonction en série, résoudre des équations différentielles, etc.

Algèbre linéaire. Maple est particulièrement doué pour l'algèbre linéaire : addition, multiplication, inversion de matrices, produits scalaire et vectoriel, déterminant, transposée, résolution de systèmes, valeurs propres, base, espace des colonnes, orthogonalisation, ...

Bien Plus. Maple permet aussi de faire des statistiques, de la théorie des nombres, des graphes, et j'en passe beaucoup. En fait, le savoir mathématique de Maple est en constante évolution.

Quelques fonctions intéressantes de Maple :

sum : pour calculer des sommes
ifactor : pour factoriser en produit d'entiers
solve : pour resoudre une ou plusieurs equations
diff : pour deriver, differencier
dsolve : pour resoudre des equations differentielles
plot : pour tracer des courbes ou des solutions d'ED
fsolve : pour resoudre numeriquement les equations
rsolve : pour resoudre des equations de recurrence
int : pour integrer des fonctions
matrix : pour definir une matrice
eigenvals : pour calculer des valeurs propres
series : pour calculer la serie de Taylor
limit : pour evaluer une limite

2.2 Les différentes interfaces de Maple

L'utilisateur peut accéder aux fonctionnalités de Maple à travers des interfaces de différentes natures.

2.2.1 La feuille de travail standard (*standard worksheet*)

Cette feuille permet un accès simple et rapide à Maple à l'aide de palettes, de menus contextuels ou encore de complétion automatique des mots-clés du langage. Le débutant y trouvera des outils qui l'aideront à formuler des problèmes simples. Par ailleurs, mais à la condition d'avoir une bonne connaissance des possibilités qu'offre Maple, cette interface permet de mettre en page et de présenter un document électronique dans lequel les calculs sont mis en valeur au moyen d'outils spécifiques, en masquant certains calculs intermédiaires, en intégrant des figures sur la feuille de travail, et en offrant la possibilité de faire figurer sur la feuille des calculs interactifs.

2.2.2 La feuille de travail classique (*classic worksheet*)

Cette interface est le mode avec interface graphique le plus ancien. S'il ne présente pas tous les gadgets disponibles dans le mode standard, il dispose néanmoins d'une interface graphique puissante ainsi que du *pretty-printer* qui affiche des réponses avec un rendu utilisant les notations mathématiques usuelles. Il présente à nos yeux l'avantage de séparer clairement les interventions de l'utilisateur de celles de l'interpréteur. Il sera souvent plus simple de développer des programmes avec cette interface, même s'il faut les intégrer ensuite dans une application beaucoup plus sophistiquée réalisée avec le mode standard. Autre avantage de cette interface : elle requiert moins de mémoire que l'interface standard.

2.2.3 La ligne de commande (*command line*)

Cette interface est un mode dépouillé, en particulier dépourvu d'interface graphique. Elle date des premières versions de Maple et permettait alors de répondre aux limitations de la mémoire. Ce mode subsiste dans les versions actuelles. Avec lui, on résout des problèmes dont la taille interdit qu'ils soient traités dans l'un des modes précédemment évoqués. Il autorise aussi une utilisation de Maple dans un traitement par lot (*batch*) avec des scripts. Avec ces trois interfaces, on accède à la plupart des commandes Maple et donc aux véritables capacités du calcul formel. Il existe encore deux autres interfaces aux buts très spécifiques. On ne peut pas les mettre sur le même plan que les trois précédents.

2.2.4 Le calculateur graphique (*Maplesoft graphing calculator)*

Le calculateur graphique est une interface qui permet d'effectuer des calculs simples ainsi que des représentations graphiques interactives. Cette interface met donc à disposition une sorte de super calculatrice graphique.

2.2.5 Les Maplets *(Maplet applications)*

Les Maplets sont des interfaces graphiques contenant des fenêtres, des champs ou encore des curseurs qui permettent de lancer des calculs et des représentations graphiques par simples clics de souris, sans avoir à accéder et à manipuler une feuille de travail. Les Maplets sont la partie visible de programmes ou d'objets Maple dont les structures sont masquées dans la présentation. Les Maplets sont apparus dans la version 8 de Maple.

2.3 Premiers pas avec Maple

2.3.1 L'invite

Après avoir lancé le logiciel Maple à travers l'interface classique, on obtient une feuille de travail vierge sur laquelle apparaît l'invite (*prompt*) qui, par défaut, est représentée par le symbole « > ».

2.3.2 Caractères de fin de saisie

Lorsque l'on utilise Maple en interagissant directement avec l'interpréteur (utilisation dite au *toplevel* par opposition à la simple exécution d'un programme Maple), l'interaction avec le logiciel consiste en une succession d'instructions saisies par l'utilisateur et de réponses fournies par l'interpréteur. La feuille de travail présente les traces de cette interaction.

L'utilisateur doit donc saisir une instruction à côté de l'invite et pour signifier la fin de sa saisie, il lui faut terminer la ligne de commande par le caractère ";" ou par le caractère ":". L'instruction saisie sera interprétée et exécutée (*si la syntaxe est correcte*) dès que l'utilisateur aura pressé la touche Entrée. Lorsqu'une instruction se finit par ";", l'interpréteur de Maple exécute l'instruction et, si elle est syntaxiquement valide, affiche le résultat obtenu :

```
> 2+3;
                              5
> exp(1.0);
                          2.718281828
```

En revanche, si l'instruction est terminée par ":", Maple exécute l'instruction de façon muette, c'est-à-dire n'affiche pas le résultat :

```
> 2+3 :
> exp(1.0) :
```

Cette façon de procéder est utile pour éviter l'affichage de résultats volumineux comme les contenus de packages ou les objets graphiques sous-jacents aux représentations graphiques.

2.3.3 Groupement d'instructions

Pour saisir plusieurs instructions sur une même ligne et ainsi les passer simultanément à l'interpréteur, il suffit de les séparer par des ";" ou des ":". Elles sont alors exécutées consécutivement, dans l'ordre où elles ont été présentées, et les résultats, s'il y a lieu, sont affichés l'un sous l'autre en une seule fois par l'interpréteur.

```
> 1+2; 2.0^(1/2); (2*3)^2;
                              3
                          1.414213562
                              36
```

On peut aérer la présentation des instructions et passer à la ligne entre chacune d'elles plutôt que de les écrire sur la même ligne. Pour passer à la ligne au sein du même crochet d'exécution sans provoquer l'exécution individuelle d'une instruction au moment du changement de ligne, il faut presser simultanément les touches **Majuscule** et **Entrée** (**Shift** et **Enter** sur un clavier anglo-saxon). On remarque qu'au moment où l'on change de ligne, le crochet d'exécution qui précède l'invite s'allonge de manière à englober les différentes lignes d'instructions.

Ce repère visuel a pour but de mettre en évidence l'ensemble des instructions adressées simultanément à l'interpréteur lors de la prochaine validation.

```
> 1+2;
```

2.0^(1/2);
(2*3)^2;
3
1.414213562
36

Cette façon d'aller à la ligne sans déclencher l'interpréteur est particulièrement utile lorsqu'on programme avec Maple. Elle permet d'écrire des fonctions Maple dont le contenu s'étend sur plusieurs lignes.
Pour écrire une instruction dont la longueur est supérieure à celle d'une ligne sur la feuille de travail, on peut formater la saisie en plaçant un caractère "\" chaque fois que l'on va à la ligne. Maple utilise aussi ce caractère pour signaler des réponses qui tiennent sur plusieurs lignes.

2.3.4 Commentaires

Le caractère "#" permet d'introduire un commentaire dans une feuille de travail. Ce qui se trouve entre ce caractère et la fin de ligne n'est alors pas pris en compte par l'interpréteur, comme le montre l'exemple suivant :

> 1+2; # un premier calcul très simple
2.0^(1/2); cette ligne n'est pas prise en compte
(2*3)^2;
3
36

2.3.5 Enregistrer son travail

Comme dans la plupart des environnements logiciels présentant des menus, on a recours à la rubrique *Save as* (ou *Save* si l'on sauvegarde de manière incrémentale un document déjà enregistré) du menu *File* pour sauvegarder son document dans un fichier (ici la feuille de travail) pour une utilisation future. Pour retrouver sa feuille de travail enregistrée dans un fichier, on l'ouvre en choisissant la rubrique *Open* du menu *File* et en indiquant le nom du fichier dans lequel est sauvegardée la feuille considérée. On récupère la feuille de travail dans l'état où elle se trouvait au moment de son enregistrement. Toutefois, il ne s'agit que d'un effet d'affichage ; les instructions qui figurent à l'écran n'ont pas été exécutées par l'interpréteur à l'instant où la feuille de travail vient d'être ouverte. Si l'on souhaite que certains résultats soient présents dans la mémoire de travail, il faut expressément provoquer l'exécution des instructions correspondantes.

2.4 Variables et affectation

Le lecteur l'aura compris, effectuer des calculs élaborés demande de conserver des résultats intermédiaires afin d'y accéder ultérieurement, d'où le besoin de variables. Une variable est un emplacement en mémoire auquel est attachée une étiquette (*le nom de la variable*) qui permet d'accéder à son contenu. L'affectation assigne une valeur à une variable ; elle établit une correspondance entre une étiquette et une valeur.

2.4.1 Noms de variables

Un nom de variable licite est un mot qui commence par une lettre suivie d'un nombre fini de caractères, lettres, chiffres ou "_" (*blanc souligné*), autres que les caractères qui jouent un rôle particulier dans le langage ("%", "#", " ?", "&", ":", ";", "=", "$". . .) de sorte que l'ensemble forme un mot n'apparaissant pas dans les mots réservés ou les mots protégés du langage Maple

```
> abc; a1234; ABC; # sont des noms de variables licites
                              abc
                             a1234
                              ABC
```

Par ailleurs, Maple distingue les majuscules des minuscules ; ainsi, a et A ne désignent pas la même variable comme le montre l'exemple suivant :

```
> a:=10;
                          a := 10
> a;
                             10
> A; A
```

2.4.2 Affectation

L'opérateur d'affectation est désigné par ":="
L'instruction nom_de_variable:=expression; construit une case mémoire dont l'étiquette est nom_de_variable et dont le contenu est le résultat de l'évaluation de l'expression. La simple instruction « nom_de_variable ; » provoque l'affichage du contenu de cette variable :

```
> a:=10+2*4; b:=x+y+2*x;
                          a := 18
                       b := 3 x + y
> a; b;
                             18
                          3 x + y
```

Il convient de connaître la règle fondamentale suivante : pour Maple, une variable vide s'évalue en son nom. En d'autres termes, le résultat de l'évaluation d'une variable non affectée est le nom de cette variable.

```
> toto;
                            toto
```

2.4.3 Opérateur *dito*

Le symbole "%" désigne l'opérateur *dito* qui permet d'obtenir le dernier résultat calculé par l'interpréteur (lorsqu'il existe) :

```
> a:=1+2*3;
                          a := 7
> %;
                             7
```

De même, les opérateurs "%%" et "%%%" permettent d'obtenir l'avant dernier et l'antépénultième[9] résultats calculés par l'interpréteur (lorsqu'ils existent) :

```
> a:=2*3+4;b:=3*4+5;c:=4*5+6;
                          a := 10
                          b := 17
                          c := 26
```

```
> %%%;
    10
```

2.4.5 Nettoyage de la mémoire de travail

Lors d'une session assez longue, il peut s'avérer judicieux de nettoyer complètement la mémoire de travail afin de récupérer de l'espace mémoire et de reprendre proprement les définitions des objets et variables utilisés. Une telle réinitialisation s'obtient avec la commande **restart** :

```
> a:=1; b:=2; a; b;
                    a := 1
                    b := 2
                      1
                      2
> restart: a; b;
                      a
                      b
```

2.5 Utilisation de l'aide en ligne

Il y a différentes façons d'utiliser l'aide en ligne mise à disposition par Maple.

2.5.1 L'opérateur " ?"

Lorsque l'on travaille avec l'interface classique, le plus simple consiste à consulter une page d'aide portant sur une commande donnée en l'activant à l'aide de l'opérateur " ?". Par exemple, si l'on souhaite connaître le contenu de la page d'aide en ligne consacrée à la commande **evalf**, on va écrire directement ?**evalf** à côté de l'invite puis valider l'instruction, sans terminer celle-ci par un ";" ou un ":".
Une page d'aide en ligne comporte différents renseignements sur la commande présentée. Elle commence par son nom et la forme théorique que doit prendre un appel utilisant la commande en question, ainsi qu'une description des paramètres d'appel. Dans une deuxième partie, cette page décrit les fonctionnalités implémentées par la commande en question ainsi que les caractéristiques, cas particuliers ou limites d'utilisation. Une troisième partie présente des exemples que l'on peut copier et coller dans sa propre feuille de manière à les adapter à son propre problème. Enfin, la page d'aide propose une série de mots-clés. Ces termes, en rapport avec la commande présentée, sont des liens cliquables qui dirigent vers d'autres pages d'aide.

Résumé des caractères spéciaux. Ils serviront à gérer la conversation avec Maple. Ils ne font pas partie des commandes au sens strict. Le tableau suivant énumère ces "méta-caractères".

Caractère	Signifie
;	J'ai termine et je veux que tu affiches la réponse
:	J'ai termine et je ne veux pas que tu affiches la réponse
%	Prends le résultat précédent
%%	Prends l'avant-dernier résultat
?	sujet information à-propos de...
#	J'inscris un commentaire

2.6 Les bibliothèques de fonctions (packages)

Lorsqu'on lance Maple, débutant ainsi une nouvelle session, seul le noyau de Maple est effectivement chargé en mémoire. Ce noyau comporte les commandes fondamentales, celles qui sont d'utilisation courante. La plupart de ces fonctions, dites *built-in,* consistent en des exécutables (généralement du code C compilé) afin de privilégier la vitesse d'exécution.

Les autres commandes disponibles sont regroupées en bibliothèques de fonctions appelées packages. Afin d'éviter de remplir inutilement l'espace mémoire de travail, les commandes des packages ne sont pas chargées lors du démarrage d'une session, mais seulement à la demande.

Il y a trois façons d'invoquer une commande figurant dans un package.

- Chargement de l'ensemble du package : La première façon consiste à charger le package dans son ensemble à l'aide de la commande **with**. Ainsi dans l'exemple suivant, on charge le package LinearAlgebra qui met à disposition de l'utilisateur des commandes relatives à l'algèbre linéaire :

> ***with***(**LinearAlgebra**);

[&*x*,*Add*,*Adjoint*,*BackwardSubstitute*,*BandMatrix*,*Basis*,
BezoutMatrix,*BidiagonalForm*,*BilinearForm*,
CharacteristicMatrix,*CharacteristicPolynomial*,*Column*,
ColumnDimension,*ColumnOperation*,*ColumnSpace*,
CompanionMatrix,*ConditionNumber*,*ConstantMatrix*,
ConstantVector,*Copy*,*CreatePermutation*,*CrossProduct*,
DeleteColumn,*DeleteRow*,⋯*VectorMatrixMultiply*,*VectorNorm*,
VectorScalarMultiply,*ZeroMatrix*,*ZeroVector*, *Zip*]

On lit le nom de toutes les commandes qui viennent d'être chargées. Dans l'exemple précédent, nous avons coupé la (longue) réponse de Maple et remplacé plusieurs lignes de la réponse par des points de suspension. On recourra au ":" plutôt qu'au ";" en fin d'instruction pour ne pas voir figurer cette liste dans la feuille de travail. Le chargement d'un package peut s'accompagner d'un message d'avertissement lorsqu'une commande du package qui vient d'être chargé porte un nom identique à celui d'une commande du noyau ou à celui d'une commande déjà chargée par ailleurs. Toutes les commandes qui viennent d'être listées sont dorénavant accessibles directement, c'est-à-dire par invocation directe de leur nom :

> A:=**Matrix**(3,3,[1,2,3,1,0,0,0,1,0]);

A := [1 2 3; 1 0 0; 0 1 0]

> **Rank**(A);

3

On peut décharger un package à l'aide de la commande **unwith** pour libérer de l'espace dans la mémoire de travail de Maple :

> **unwith**(**LinearAlgebra**);

> **Rank**(A);

Rank(*A*)

Une fois le package **LinearAlgebra** déchargé, la commande **Rank** n'est plus accessible c'est pourquoi Maple répond en répétant l'instruction soumise à l'interpréteur.

- Chargement de la commande : une deuxième façon, plus économique en terme de mémoire, d'accéder à une commande qui se trouve dans un package consiste à ne charger que la commande concernée, toujours à l'aide de la commande **with**, en passant son nom en deuxième argument :

```
> with(LinearAlgebra,Rank);
                          [Rank]
> A:=Matrix(3,3,[1,2,3,1,0,0,0,1,0]);
                          [1  2  3]
                     A    [1  0  0]
                          [0  1  0]
> Rank(A);
                            3
```

- Invocation par le nom long : la troisième façon de procéder pour accéder à une commande figurant dans un package consiste à préciser, au moment de l'appel, le nom du package dans lequel figure la commande :

```
> A:=Matrix(3,3,[1,2,3,1,0,0,0,1,0]);
                          [1  2  3]
                     A    [1  0  0]
                          [0  1  0]
>LinearAlgebra[Rank](A);
                            3
```