Premiers pas avec MATHEMATICA

LP206 : Mathématiques pour physiciens I

Année 2010/2011

1 Introduction

MATHEMATICA est un logiciel de calcul formel qui permet de manipuler des expressions mathématiques symboliques. Contrairement à la plupart des langages de programmation (tels que C et FORTRAN), il peut manier non seulement des quantités numériques (nombres entiers, réels ou complexes), mais aussi des polynômes, des fonctions, des séries, etc.

MATHEMATICA permet d'effectuer des opérations telles que la dérivation, l'intégration, le calcul de limites, ainsi que la simplification d'expressions compliquées et la résolution d'équations. Grâce à l'interface graphique, il est également possible de visualiser les résultats en traçant des courbes ou même des surfaces dans l'espace tridimensionnel.

La version standard de MATHEMATICA contient des milliers de fonctions prédéfinies. Il est possible d'augmenter encore les possibilités en incluant des bibliothèques de fonctions adaptées à des utilisations plus spécifiques dans des domaines spécialisés de physique, mathématiques, chimie, ingénierie, etc. De ce fait, MATHEMATICA est devenu un outil de travail quotidien pour bon nombre de chercheurs.

2 Utilisation de base

En LP206, nous allons utiliser la version graphique de MATHEMATICA, qui est la plus simple.[1]

[1] Il existe aussi une version textuelle qui peut s'avérer utile lors de l'utilisation d'un ordinateur distant par l'intermédiaire d'un client SSH. Il est en outre possible d'appeler MATHEMATICA à l'aide de commandes en un langage de type *shell* sous UNIX, ce qui permet d'intégrer les calculs en MATHEMATICA dans des automatismes massifs faisant intervenir, par exemple, d'autres langages de programmation.

Pour bien profiter de cette introduction, il ne suffit pas de la lire. Il faut au contraire ouvrir une session MATHEMATICA *et retaper les exemples de commandes pour voir comment les choses se passent en pratique. N'hésitez pas à faire vos propres expériences en essayant quelques variantes. Vérifiez à chaque étape que vous comprenez parfaitement le résultat obtenu par* MATHEMATICA.

Lors du démarrage de MATHEMATICA, on se retrouve avec une feuille de calcul vierge. On peut alors écrire une ligne de commande qu'on fera exécuter en tapant [*Majuscule* + *Entrée*].

En voici quelques exemples simples :

```
2 3 4  ;[2]
(3+4)^2-2(3+1)/4  .
```

On notera que le circonflexe indique la puissance et que l'espace entre deux nombres sous-entend un signe de multiplication (on peut aussi utiliser un astérisque). Les espaces dans d'autres positions sont simplement ignorés (on peut en mettre si cela rend l'expression plus lisible).

Sur une feuille de calcul compliquée, on a intérêt à ajouter des commentaires, soit pour se souvenir ultérieurement de la logique du calcul, soit pour le présenter à autrui :

```
(* On calcule le nombre de choix de 13 objets parmi 30. *)
30!/(13! 17!)  .
```

Les calculs en MATHEMATICA sont en général faits avec une précision arbitraire. Par exemple, les commandes

```
2^100
```

et

```
50!
```

donnent des entiers exacts et assez longs. Si l'on désire un résultat approximatif on peut écrire

```
N[Pi]  ;
N[2^100]  ;
N[E,50]  .
```

[2]Les symboles de ponctuation précédés d'un large espace ne font pas partie de l'instruction MATHEMATICA.

Par défaut, MATHEMATICA affiche les calculs avec 6 chiffres significatifs, quitte à passer en notation exponentielle si besoin est. On peut spécifier une précision plus élevée en donnant le nombre de chiffres significatifs en second argument.

Il faut noter que les entiers sont considérés comme des expressions exactes, tandis qu'une expression avec un point décimal[3] est une approximation numérique (par défaut correcte à 6 chiffres). Si l'on mélange plusieurs approximations numériques, MATHEMATICA ajuste la précision à celle du membre le moins précis, comme le montre cet exemple :

```
1. + N[Pi,50]  .
```

3 Les fonctions prédéfinies

MATHEMATICA possède une large collection de fonctions prédéfinies. Par convention, ces fonctions commencent par une majuscule et prennent leur argument entre des crochets. Si plusieurs arguments sont nécessaires, on les sépare par des virgules :

```
Sqrt[36]  ;
Exp[2.3]  ;
Log[E^5]  ;
Log[2,1024]  ;
Sin[3 Pi/2]  .
```

`Log[x]` est le logarithme népérien (ln) ; `Log[2,x]` le logarithme en base 2.

D'autres fonctions indispensables sont `Cos[x]`, `Tan[x]`, `ArcSin[x]`, `ArcCos[x]`, `ArcTan[x]`, `Abs[x]`, `Cosh[x]`, `Sinh[x]`, `Tanh[x]`.

4 L'aide en ligne

Étant donné le nombre de possibilités, l'aide en ligne devient vite indispensable. Il suffit de mettre le signe `?` devant le nom de la fonction pour accéder à l'aide. La variante `??` donne plus d'informations. Si on ne se souvient pas de l'écriture complète du commande, on peut remplacer les lettres inconnues par `*` :

```
?Cos  ;
??FactorInteger  ;
?Si*  .
```

Pour des informations encore plus complètes, on peut utiliser un moteur de recherche comme GOOGLE en ajoutant le mot MATHEMATICA à la recherche.

[3] MATHEMATICA utilise le point pour séparer les parties entière et décimale d'un nombre.

5 Calculs plus avancés

Il est possible de combiner des résultats obtenus précédemment pour faire des calculs plus complexes. On peut utiliser `%` pour référer au résultat précédent, `%%` à celui d'avant, et ainsi de suite. Par exemple :

```
7^2  ;
% + 1  ;
3 % + %^2 + %%  .
```

L'expression `%n` réfère au résultat `Out[n]`.

Un outil plus puissant est les variables :

```
x = 5  ;
2 x - 3 x^2  ;
coulomb2 = 1.602 10^-19  .
```

On peut choisir n'importe quel nom, à condition que le premier symbole soit une lettre. Il est recommandé de toujours commencer par une lettre minuscule pour éviter toute confusion avec les fonctions et variables prédéfinies : celles-ci commencent en effet par une majuscule. Exemples :

```
N[Pi,30]  ;
N[EulerGamma,50]  .
```

Si l'on veut réutiliser le même nom de variable `x` dans un autre contexte plus tard, on peut le libérer avec `Clear[x]`. Il est bien entendu interdit de libérer les variables prédéfinies.

Certains calculs nécessitent l'utilisation d'une variable d'itération. Par exemple, $\sum_{i=1}^{10} i^2$ peut se calculer ainsi :

```
Sum[i^2,{i,1,10}]  .
```

Les produits se calculent avec la commande `Product`.

On peut également utiliser des listes, c'est-à-dire des objets regroupés par des accolades :

```
kk = {2,3,5,7,11}  ;
Sum[i^2,{i,kk}] (* i prend ses valeurs dans l'ensemble kk *)  .
```

Il existe de nombreuses possibilités pour manipuler des listes (surtout lorsqu'elles sont imbriquées). Pour extraire un élément d'une liste on utilise des doubles crochets :

```
kk[[4]]  .
```

On peut exécuter plusieurs commandes sur une même ligne en les séparant par des points-virgules (;). La sortie d'une commande terminée par « ; » n'est pas affichée. Exemple :

```
x=4; y=6; z=y+6  .
```

6 Calculs symboliques

Une des forces de MATHEMATICA est de pouvoir manipuler des expressions symboliques, même compliquées, de manière efficace.

```
Clear[x,y]; (x-1)^10 (y-2)^5  ;
Expand[%]  ;
Factor[%]  .
```

On peut attribuer une valeur à une variable utilisée dans une expression entrée précédemment et calculer cette dernière :

```
Sum[Sin[k x],{k,1,3}]  ;
% /. x -> Pi/4  .
```

Bien noter la différence avec l'attribution d'une valeur définitive à la variable, comme dans `x = Pi/4`.

On peut simplifier des expressions compliquées ou bien les stocker dans des variables :

```
expr = ((1+Sqrt[5])/2)^10  ;
Simplify[expr]  ;
Sin[x]^2 + Cos[x]^2  ;
Simplify[%]  .
```

Il est possible de comparer une expression avec une autre :

```
Sin[x]^2 + Cos[x]^2 == 1  ;
Simplify[%]  .
```

Bien noter la différence entre les écritures « `=` » et « `==` » : La première est une attribution et la seconde une comparaison.

MATHEMATICA peut résoudre des équations. Il n'est pas nécessaire de spécifier pour quelle variable on veut les résoudre, sauf s'il y en a plusieurs :

```
Solve[21 x^2 - 68 x + 55 == 0,x]  ;
Solve[x^3 - 2 x^2 + x - 1 == 0]  ;
Simplify[%]  ;
FullSimplify[%]  .
```

Parfois il n'est pas possible de résoudre une équation en termes de fonctions connues, ou bien la résolution est possible mais donne une expression très compliquée. On peut alors chercher des solutions numériques :

```
Solve[x^5 + x + 1 == 0,x]  ;
NSolve[x^5 + x + 1 == 0,x]  .
```

Cette stratégie échoue si l'équation est transcendantale.[4] On peut alors trouver numériquement une racine près d'une valeur initiale spécifiée :

```
eqn3 = 3 Cos[x]==Log[x]  ;
NSolve[eqn3,x]  ;
FindRoot[eqn3,{x,1}]  ;
FindRoot[eqn3,{x,10}]  .
```

7 Nombres complexes

Il est possible de faire des calculs avec des nombres complexes :

```
z = (4 + 3I)/(2 - I)  ;
Re[z]  ;
Im[z]  ;
N[Exp[2 + 9I]]  .
```

Si $z = a + \mathrm{i}\, b$, alors le conjugué $\bar{z} = a - \mathrm{i}\, b$ est obtenu par `Conjugate[z]`. Dans l'écriture $z = |z|\, \mathrm{e}^{\mathrm{i}\,\phi}$, la norme $|z|$ est donnée par `Abs[z]` et l'argument ϕ par `Arg[z]`.

Dans les manipulations courantes, MATHEMATICA suppose que tous les nombres peuvent être complexes. Il est cependant possible de développer une expression complexe en imposant aux variables des valeurs réelles. Comparer

```
Sin[x + I y]  ;
Expand[%]  ;
ComplexExpand[%]  .
```

MATHEMATICA connait bien sûr la formule de Moivre, mais développe le résultat avec beaucoup de précautions :

```
moivre = (Cos[x] + I Sin[x])^n  ;
FullSimplify[moivre]  ;
ComplexExpand[%]  .
```

[4] Une équation algébrique est du type *polynôme*(*racine*) = 0. Une équation transcendantale est une équation qu'on ne peut ramener à une expression de ce genre.

8 Différentiation et intégration

MATHEMATICA peut dériver une fonction par rapport à une de ses variables (dérivée partielle) :

```
D[Sqrt[Tanh[x]],x]  ;
D[ArcTan[x y],x]  .
```

Les différentielles totales se calculent ainsi :

```
Dt[x^n]  ,
```

où la différentielle dx est notée `Dt[x]`.

Les intégrales indéfinies (= primitives) se calculent en un tour de main :

```
Integrate[1/(x^4-a^4),x]  .
```

Il en est de même pour les intégrales définies (propres ou impropres) :

```
Integrate[Tan[x],{x,0,Pi/4}]  ;
Integrate[Sin[x]/x,{x,0,Infinity}]  .
```

Idem pour les intégrales multiples :

```
Integrate[x^2+y^2,{x,0,1},{y,0,x}]  .
```

L'intégration se fait d'abord sur la variable y, puis sur la variable x.

Parfois le résultat nous permet de faire connaissance avec des fonctions exotiques :

```
Integrate[Log[1+Tanh[x]],{x,0,Pi}]  ;
?PolyLog  .
```

9 Sommes et produits

Les sommes finies et les séries se calculent aisément :

```
Sum[x^i/i,{i,1,7}]  ;
Sum[1/k^2,{k,1,Infinity}]  ;
Sum[x^n/n,{n,1,Infinity}]  .
```

Que peut bien vouloir dire l'expression

```
Sum[1/x^4,{x,1,Infinity,2}]  ?
```

(*Indication* : utiliser l'aide en ligne !)

Les produits se calculent de la même manière :

```
Product[(x-a),{a,1,10}]  ;
prod = Product[(3j-1)^2/((3j)(3j-2)),{j,1,Infinity}]  .
```

Il est (peut-être) rassurant de voir que MATHEMATICA ne connaît pas *toutes* les règles de simplification algébrique :

```
FullSimplify[prod == 3 Gamma[1/3]^3 / (4 Pi^2)]  ;
N[%,100]  .
```

10 Fonctions et programmes

Nous pouvons définir nos propres fonctions :

```
poly[x_] := 3 - 2 x + 4 x^2  ;
poly[4]  ;
f[x_,y_] := x Sin[y^2 / x]  ;
f[2,Sqrt[Pi]]  .
```

Faites bien attention à la notation : l'écriture utilisée pour une définition (`:=`) est différente de celle employée pour une attribution (`=`). Le tiret bas du côté gauche indique qu'on définit ce par quoi la variable en question sera remplacée du côté droit. On peut comparer avec

```
1 + f[x] + f[y] /. f[t_] -> t^2  .
```

Pour ceux qui sont familiers avec un langage de programmation, l'écriture `f[x_] :=` invoque une *définition* d'une fonction ou procédure, tandis que `f[x]` est un *appel* de cette même fonction.

MATHEMATICA fournit de nombreuses fonctionnalités pour éviter un travail répétitif. En particulier, on peut faire des boucles,

```
Do[Print[n!],{n,1,10}]  ,
```

ainsi que des structures de contrôle,

```
Do[If[Mod[n,7]!=0 && Mod[n,5]!=1,Print[n]],{n,1,50}]  .
```

Cette instruction énumère les nombres entre 1 et 50 qui ne finissent pas par un « 1 » ou un « 6 » et qui ne sont pas divisibles par 7.

11 Séries de Taylor

La série de Taylor d'une fonction $f(x)$ autour de x_0 à l'ordre n s'obtient en général par

```
Series[f[x],{x,x0,n}]  .
```

Quelques examples :

```
dev1 = Series[Exp[x],{x,0,5}]  ;
dev2 = Series[(1+x)^n,{x,0,2}]  .
```

Noter que MATHEMATICA garde la précision du développement au moyen de la notation $O(x^n)$. De même que pour les approximations numériques, la combinaison d'objets de précisions différentes est faite de manière cohérente :

```
dev1 + dev2  .
```

La fonction `Normal[...]` élimine la notation $O(x^n)$ et permet de traiter le résultat comme une expression ordinaire par la suite :

```
Normal[dev1 + dev2]  ;
Solve[%==0,x]  .
```

12 Limites

Voici d'abord deux tentatives infructueuses de trouver la limite $\lim_{x\to 0} \frac{\sin x}{x}$, suivies par la solution qui marche :

```
t = Sin[x]/x  ;
t /. x->0  ;
t /. x->0.01  ;
Limit[t,x->0]  .
```

13 Équations différentielles

Définissons d'abord une équation différentielle et sa condition initiale :

```
equadif = y'[x] == a y[x]  ;
condinit = y[0] == 1  .
```

La solution générale et celle qui satisfait la condition initiale s'obtiennent alors par

```
DSolve[equadif,y[x],x]
```

et

```
DSolve[{equadif,condinit},y[x],x]  .
```

Noter que le deuxième argument est la fonction pour laquelle on veut résoudre l'équation différentielle, tandis que le troisième argument est la variable indépendante.

Il est également possible de résoudre des équations différentielles couplées :

```
DSolve[{x'[t]==y[t], y'[t]==x[t]}, {x[t],y[t]}, t]  .
```

14 Graphes

Tracer le graphe d'une fonction ou d'une collection de fonctions est aisé avec MATHEMATICA :

```
Plot[Sin[x],{x,0,2 Pi}]  ;
Plot[{Sin[2 x],Cos[3 x]},{x,0,2 Pi}]  .
```

Les possibilités pour personnaliser le graphe sont nombreuses. Exemple :

```
Plot[Sin[x^2],{x,0,3},Frame->True,GridLines->Automatic]  .
```

Vous pouvez consulter l'aide en ligne pour plus de détails.

Pour une fonction de plusieurs variables, il peut être utile de visualiser ses courbes de niveau :

```
ContourPlot[Sin[x] Sin[y], {x,-2,2}, {y,-2,2}]  .
```

On peut également faire des graphes paramétriques en trois dimensions. Par exemple, une hélice et un tore sont produits respectivement par

```
ParametricPlot3D[{u Sin[t], u Cos[t], t/3}, {t,0,15}, {u,-1,1}]
```

et

```
ParametricPlot3D[{Cos[t](3+Cos[u]), Sin[t](3+Cos[u]), Sin[u]},
    {t,0,2 Pi}, {u,0, 2 Pi}]  .
```