Cours-TD n° 2 : Éléments de langage Mathematica

Table des matières

1 Préparations pour ce TD

1.1 Raccourcis clavier

N'oubliez pas le feuille avec les **raccourcis clavier** et essayez d'utiliser la souris le moins possible.

1.2 Exemples de code Mathematica

Pour bien apprendre les notions traitées ici, il est très imortant de tester **et comprendre** les exemples de code Mathematica donnés ici. Il ne suffit pas juste de taper ces lignes de code sans chercher à les comprendre.

Tous les exemples de code Mathematica montrés dans l'énoncé ne contiennent que des cellules de type Input. C'est pour cela que c'est indispensable de tester ces exemples soi-même avec Mathematica. **Chaque ligne d'un exemple est une nouvelle cellule** de type Input et doit être évaluée dans l'ordre. Les exemples peuvent être copié&collé (Ctrl+C puis Ctrl+V) dans Mathematica depuis la version PDF de l'énoncé présent, mais attention de séparer la commande Quit dans une cellule à part. En général si les commandes ne sont pas trop longues, il est conseillé de ne pas utiliser le copier&coller, mais de taper soi-même les commandes. Cela aide à l'apprentissage de ces commandes et de plus le copier&coller ne fonctionne pas dans certains cas.

1.3 Exercices

Les exercices sont indiqués avec un numéro sur le bord gauche de l'énoncé. Pensez à indiquer ce numéro dans votre notebook d'une manière bien visible, par exemple en utilisant le type de cellule "Section" (Alt+4).

1.4 Pour aller plus loin

Ceux qui auront déjà terminé tous les exemples et exercices avant la fin de la séance peuvent étudier quelques "Problem" au choix du livre de Hazrat (voir sur SAKAI sous le dossier "livres") et d'essayer de résoudre quelques "Exercice" (solutions à la fin du livre).

2 Wolfram Alpha, langage naturel et Step-by-step

2.1 Wolfram Alpha

Wolfram Alpha (www.wolframalpha.com) est un outil en ligne gratuit répondant à toutes sortes de questions (qui doivent être posées en anglais) et qui utilise une grande base de donnée et des calculs Mathematica. Il permet d'accéder avec un simple navigateur internet à beaucoup de fonctionnalités de Mathematica. Par exemple, la demande "derivative of sin(x)" (dérivée de sin(x)) calcule la réponse "cos(x)" et donne au passage d'autres informations utiles (propriétés de la fonction, courbes, développements limités, etc...). A essayer !

1. Tester sur Wolfram Alpha les requêtes suivantes :
- ▷ derivative of sin(x)
- ▷ Plot sinus from 0 to 10
- ▷ UPMC

2.1.1 Wolfram Alpha depuis Mathematica

Wolfram Alpha peut être interrogé directement depuis Mathematica avec une connexion Internet.

2. Dans un nouveau notebook (Ctrl + N), formuler une requête à Wolfram Alpha avec un double signe égal (ne pas copier&coller, cela ne marche pas ici !) :
`== GDP France`
Si un message d'erreur apparait c'est que la connexion Internet n'est pas bien configurée. Sur les postes de l'UTES ou le bureau à distance de l'UTES il faut configurer Mathematica pour qu'il puisse accéder à Internet : Dans Mathematica aller dans le menu **Edit->Preferences** puis dans l'onglet **Internet Connectivity** choisir l'onglet **Proxy Settings** puis cocher la case **Use the following settings** et entrer **proxyweb.upmc.fr** dans la case **HTTP Proxy** et 3128 dans la case **Port**. Tester la connexion Internet avec le button **Test Internet Connectivity**.

2.2 Langage naturel

Depuis l'introduction de Wolfram Alpha (www.wolframalpha.com), Mathematica peut aussi être utilisé en langage naturel, c'est-à-dire des instructions écrites avec des phrases en anglais. Voici un exemple :

Au début d'une cellule "input" taper = puis formuler votre instruction en langage naturelle (en anglais uniquement), par exemple : = Plot sinus from 0 to 10

Si on ne se rappelle plus de la syntaxe d'une fonction en Mathematica, on peut soit chercher dans l'aide (F1), soit essayer de décrire ce qu'on veut faire en anglais après un signe = depuis Mathematica. Avec un peu de chance Wolfram Alpha comprend votre demande et ne donne non seulement la solution, mais aussi la traduction de votre demande en syntaxe Mathematica.

2.3 Step-by-step

Wolfram Alpha Pro (la version payante de Wolfram Alpha) a introduit une nouvelle fonctionnalité : le *step-by-step :*

http://www.wolframalpha.com/pro/step-by-step-math-solver.html

Le step-by-step permet d'obtenir une déduction mathématique avec toutes les étapes intermédiaires de presque n'importe quel problème, comme par exemple :

- ▷ Résolution d'un système d'équations
- ▷ Calcul d'une dérivée ou d'une primitive
- ▷ Transformations algébriques et trigonométriques (moins évident, à tester)

Ceci est une petite révolution, car Mathematica ne permettait jusqu'ici que d'avoir le résultat final, sans donner automatiquement les étapes intermédiaires. Grâce à l'intégration de Wolfram Alpha dans Mathematica, on peut avoir ces étapes intermédiaires directement dans un notebook et ceci sans payer un abonnement pour la version Pro de Wolfram Alpha ! Pour cela on peut faire appel à la fonction `WolframAlpha` avec une petite fonction `ShowSteps` qui filtre les résultats *step-by-step* :

```
ShowSteps[exp_] :=   WolframAlpha[ToString@HoldForm@InputForm@exp,
TimeConstraint -> Infinity,
IncludePods -> {"Input", "Indefinite Integral", "Root", "Result", "Limit"},
PodStates -> {"Step-by-step solution", "Show all steps"}]

SetAttributes[ShowSteps, HoldAll]
```

Ceci est une version améliorée de la fonction `ShowSteps` trouvée ici :

http://mathematica.stackexchange.com/questions/148/get-a-step-by-step-evaluation-in-mathematica

Sans rentrer dans les détails de cette fonction, il suffit de copier&coller cette fonction dans son notebook puis l'évaluer avec Maj+Entrée. Vous trouvez la fonction `ShowSteps` aussi dans le fichier `showsteps.nb` sur SAKAI.

3. On peut vérifier, si la fonction `ShowSteps` est bien définie avec la commande `?ShowSteps`, puis en la testant avec par exemple :

```
D[Sin[x^2], x] // ShowSteps
```

3 Définition et utilisation de fonctions

3.1 Introduction

On avait utilisé dans le dernier TD l'exemple `y:=x+2`, qui définit une dépendance de `y` par rapport à `x`. Pour indiquer une dépendance clairement, on utilise plutôt des fonctions avec des arguments. Pour cet exemple on écrit :

```
Quit
f[x_] := x+2
```

On remarque le même signe := pour la définition, c'est-à-dire la fonction est seulement définie à ce moment sans être évaluée pour une valeur donnée. C'est pour cela qu'on n'obtient pas de cellule **Out** ici.

Pour l'évaluer, on fait appel à la fonction en lui donnant une valeur pour `x_` :

```
f[2]
```

Dans la définition de la fonction, il faut faire attention à quatre particularités de syntaxe :

1. Utiliser := de préférence. Même si = peut fonctionner aussi, il y a des cas où uniquement := est adapté. Voir section 3.2 pour plus de détail.
2. La séquence des arguments est entourée avec des crochets [ ]
3. Chaque nom d'argument doit être terminé avec un tiret bas `x_` dans la séquence des arguments. Ce tiret bas indique que l'argument peut prendre n'importe quel nom d'objet et n'importe quel type d'objet. Mathematica indique avec la couleur verte qu'il s'agit d'un argument d'une fonction. On dit aussi que c'est une variable *muette*.
4. Le nom de la fonction doit être conforme aux mêmes règles que pour les variables (voir en haut), notamment commencer avec une lettre minuscule pour éviter des confusions avec les fonctions intégrées dans Mathematica.

Sinon pour vérifier qu'une fonction `f` est bien définie, on utilise `?f`. Si la définition n'est pas bonne, alors on doit d'abord effacer la définition existante de la fonction avec `Clear[f]`, puis redéfinir la fonction correctement.

Essayer cet exemple qui contient une erreur de syntaxe (mettre chaque ligne dans une cellule séparée !) :

```
Quit
f[x] := x^2
f[4]
```

Si on obtient dans “Out” la même chose que dans “In”, alors il y a probablement une erreur.

4. Après avoir testé les lignes de l'exemple au-dessus, vérifier la définition de `f` puis redéfinir celle-ci correctement (ne pas utiliser `Quit` entre les deux !).

Voici un exemple d'une fonction avec deux arguments :

```
Quit
f[x_, y_] := Sqrt[x^2 + y^2]
f[1, 1]
```

Il est très simple en Mathematica de faire des fonctions composées :

```
Quit
f[x_] := x^2 + 2
g[x_] := Sin[x] + Cos[x]
f[f[x]]
f[g[x]]
g[f[x]]
```

5. Définir $f(x) = \sqrt{1+x}$ dans Mathematica et montrer que :

$$f(f(f(x))) = \sqrt{1+\sqrt{1+\sqrt{1+x}}}$$

6. En utilisant la fonction Simplify[] et une fonction composée, montrer avec Mathematica que

$$\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1+x}}}} = \frac{3+2x}{5+3x}$$

3.2 = ou :=

Comme avec les variables, on peut définir une fonction soit avec =, soit avec :=

Pour les variables on avait déjà vu que :

- ▷ *lhs* = *rhs* *rhs* est évalué immédiatement et le résultat est copié à *lhs*
- ▷ *lhs* := *rhs* *rhs* est évalué à chaque fois que *lhs* est utilisé

Pour bien choisir entre les deux, voici quelques règles générales pour les fonctions :

- ▷ Choisir *lhs* = *rhs* si *lhs* est la “valeur finale” de *rhs* (exemple : `f[x_]=1-x^2`)
- ▷ Choisir *lhs* := *rhs* si *rhs* donne une “commande” ou un “programme” à exécuter à chaque fois qu'on demande la valeur de *lhs* (exemple : `f[x_]:=Expand[1-x^2]`)
- ▷ Choisir *lhs* := *rhs* si `f[x]` ne peut pas être évalué tant que `x` n'a pas une valeur particulière, comme pour les fonctions récursives ou les fonctions définies par morceau.
- ▷ Choisir *lhs* := *rhs* si *lhs* = *rhs* donne des messages d'erreur.
- ▷ Choisir *lhs* := *rhs* en cas de doute.

Voici un exemple pour illustrer la différence entre = et :=

```
Quit
ex[x_]:=Expand[(1+x)^2]
?ex
iex[x_]=Expand[(1+x)^2]
?iex
ex[y+2]
iex[y+2]
```

Plus d'information sur le sujet dans l'aide de Mathematica sous :
tutorial/ImmediateAndDelayedDefinitions

3.3 Notation fonctionnelle, Postfix, Prefix et Infix

On peut appeler/utiliser une fonction avec quatre notations différentes :

`f[x]`	notation fonctionnelle standard
`f@x`	notation Prefix
`x//f`	notation Postfix
`x~f~y`	notation Infix

Exemples :

```
Quit
f@x+y
f@(x+y)
x+y//f
(x+y)//f
4/6//N
{a,b,c}~Join~{d,e}
```

3.4 Fonctions anonymes ou pures

Il peut être parfois utile de définir une fonction uniquement localement dans une expression. Ceci est possible en Mathematica avec les fonctions *anonymes* ou *pures*. Ces fonctions n'ont pas de nom, mais on définit uniquement le corps de la fonction. La syntaxe est un peu particulière. Par exemple la fonction pure équivalent à $f(x) = x^2 + 4$ s'écrit :

```
(#^2 + 4)&
```

L'argument de la fonction pure est noté `#` et la fin de la définition de la fonction pure est marquée avec `&`. Pour une meilleure lecture, il est préférable d'entourer une fonction pure avec des parenthèses.

On peut appliquer cette fonction à n'importe quelle donnée en utilisant les crochets après le signe `&` :

```
(#^2 + 4)&[3]
```

7. Réécrire ces fonctions pures en fonction classique avec nom et crochets :

```
PrimeQ[#! + 1] &@4
18//2^#+#&
```

Si on a plus qu'un seul argument dans une fonction pure, on adresse ces arguments avec `#1, #2, ...`, exemple pour $f(x, y) = \sqrt{x^2 + y^2}$:

```
Sqrt[#1^2 + #2^2]&[3,4]
```

4 Listes

4.1 Introduction

Une *liste* en Mathematica est une collection ordonnée d'objets. Les objets dans une liste peuvent être de n'importe quel type et n'ont pas besoin d'être du même type, comme c'est souvent le cas dans d'autres langages de programmation. Un exemple d'une liste (accolades sur le clavier : AltGr + 4 ou AltGr + +) :

```
{4/6, x^2+2, bonjour, 1.01}
```

La syntaxe est la même que pour un ensemble mathématique, mais les listes se distinguent des ensembles par deux différences majeures :

1. L'ordre des éléments d'une liste est important, tester par exemple : `{1, 2} == {2, 1}`
2. Le même objet peut être contenu plusieurs fois dans une liste, tester par exemple : `{1, 2, 1} == {1, 2}`

4.2 Adressage d'un élément d'une liste

Pour accéder à un élément d'une liste on ne peut pas utiliser les simples crochets, comme dans le langage C, car ils sont déjà réservés pour la séquence des arguments d'une fonction. Mais la syntaxe est très similaire : on utilise des doubles crochets.

8. Tester ces exemples et répondre aux questions suivantes :
a) Quel est le premier indice d'une liste ?
b) Qu'est-ce que signifie un indice négatif ?
c) Comment est-ce qu'on accède à un élément d'une liste qui est elle-même dans une liste ?
d) Comment est-ce qu'on obtient d'une liste plusieurs éléments à la fois ?

```
Quit
liste = {4/6, x^2+2, bonjour, {a, b, c}, 1.01}
liste[[1]]
liste[[5]]
liste[[-1]]
liste[[4,1]]
liste[[4,2]]
liste[[{-2,2}]]
liste[[4,{2,3}]]
```

Pour plus d'informations sur les listes, entrer ceci dans la barre de recherche de l'aide de Mathematica : tutorial/ListsOverview

4.3 Fonctions qui génèrent des listes

Regarder l'aide de Mathematica (F1) sur la fonction `Range` pour comprendre ces exemples :

```
Range[10]
Range[3, 11]
Range[2, 17, 4]
```

Une fonction très utilisée pour générer une liste est la fonction `Table`, elle prend deux arguments :

1. Une fonction
2. Une recette similaire à la fonction `Range` pour générer une suite d'arguments pour la fonction donnée en premier argument

Exemples :

```
Quit
Table[2*n+1, {n, 1, 13}]
Table[x^i + y^i, {i, 2, 17, 4}]
```

9. La fonction `Fibonacci[i]` donne le i-ème nombre de Fibonacci. Générer une liste avec les 30 premier nombres de Fibonacci en utilisant la fonction `Table` et `Fibonacci`.

4.4 Appliquer une fonction à chaque élément d'une liste

On peut vouloir appliquer une fonction `f` à chaque élément d'une liste `{a,b,c}` pour obtenir ceci : `{f[a],f[b],f[c]}`. La plupart des fonctions Mathematica sont *listable*, c'est-à-dire on peut appliquer la fonction directement sur une liste, comme par exemple : `Sqrt[{a,b,c}]`. Toutes les fonctions arithmétiques sont *listable* :

```
1 + {a, b, c}
3 * {a, b, c}
{a, b, c} ^ 3
1 / {a, b, c}
```

10. En utilisant les fonctions `Table` et `Sqrt`, démontrer que $n(n+1)(n+2)(n+3)+1$ est toujours un nombre carré pour n entier et positif. On peut se limiter ici pour $1 \leq n \leq 10$.

Pas toutes les fonctions sont *listable*, mais on peut toujours appliquer la fonction à chacun des éléments d'une liste avec la fonction `Map` :

```
Quit
f[{a, b, c}]
Map[f, {a, b, c}]
f /@ {a, b, c}
```

Comme l'exemple montre, on peut écrire `Map` avec une notation Infix raccourcie : `/@`

4.5 Sélectionner dans une liste

On utilise ici la fonction `Select` pour sélectionner des éléments d'une liste qui satisfont certains critères. La fonction `Select[list,f]` retourne tous les éléments x_i d'une liste pour lesquelles f[x_i] donne **True**. f doit alors être une fonction booléenne, qui retourne soit True, soit False. Exemples :

```
Quit
Select[{1,2,3,4,5,6}, EvenQ]
Select[{1,2,3,4,5,6}, (# > 2)&]
```

On voit dans le deuxième exemple l'intérêt des fonctions pures (voir en haut) : définir une fonction localement pour un usage unique.

11. En utilisant les fonctions `Select, PrimeQ, Length, Table` essayer de résoudre le problème suivant : Combien de nombres de la forme $3n^5+11$ sont des nombres premiers pour
a) $1 \leq n \leq 20$
b) $1 \leq n \leq 2000$

12. En utilisant une fonction pure et les fonctions `Select, PrimeQ, Range` essayer de répondre à la question suivante : Pour quels n entre 1 et 1000 la formule 2^n+1 produit un nombre premier ?

5 Faire des graphes

5.1 Introduction

On introduit ici quelques fonctions de base pour tracer des graphes avec Mathematica. On se limite dans ce TP au cas 2D, sachant que la plupart des fonctions traitées ont leur équivalent en 3D en Mathematica.

Voici un tableau qui regroupe différents types de fonctions ou questions et les fonctions "Plot" adapté :

Fonction/Question	Exemple	Commande
$f(x)$	$\sin(x)/x$	`Plot`
$x=f(t),\ y=g(t)$	$x=\sin(3t),\ y=\cos(4t)$	`ParametricPlot`
$f(x,y)=0$	$x^4-(x^2-y^2)=0$	`ContourPlot`
$f(x,y)\geq 0$	$x^4+(x-2y^2)>0$	`RegionPlot`
$r=f(\theta)$	$r=3\cos(5\theta)$	`PolarPlot`

5.2 Plot

5.2.1 Introduction

Quand on cherche l'aide (F1) sur Plot, on obtient la séquence des ses arguments :
`Plot`[f, $\{x,\ x_{min},\ x_{max}\}$]
`Plot`[$\{f_1,\ f_2, ...\}$, $\{x,\ x_{min},\ x_{max}\}$]

1. En premier argument une fonction f ou plusieurs fonctions dans une liste
2. En deuxième argument il faut lui donner l'intervalle en x

Essayer ces exemples :

```
Quit
Plot[Sin[x], {x, 0, 2*Pi}]
Plot[Tan[x], {x, -3, 3}]
Plot[Tan[x], {x, -3, 3}, Exclusions -> Cos[x] == 0]
Plot[Tan[x], {x, -3, 3}, Exclusions -> Automatic]
Plot[Tan[x], {x, -3, 3}, Exclusions -> None]
Plot[{Sin[x], Sin[2*x]}, {x, 0, 2*Pi}]
```

5.2.2 Options de Plot

En plus des deux arguments montrés en haut, on peut changer des options dans `Plot` avec une séquence de règles :
`Plot`[f, $\{x,\ x_{min},\ x_{max}\}$, option1->value, option2->value, ...]

13. Regarder le tutoriel de Mathematica sur le sujet en ouvrant le documentation center (F1) puis dans la barre de recherche :
tutorial/Options

14. Qu'est-ce que fait l'option `Exclusions` dans l'exemple en haut ? Tracer $f(x) = 1/\sin(x)$ sans les discontinuités. Ajouter l'option `ExclusionsStyle -> Dotted`

15. (a) Sur le plot de la question précédente, ajouter des labels sur les axes. (b) Refaire un deuxième plot, mais avec un cadre (*Frame*) et une grille (*Grid*). Ici, il faut changer l'option `AxesLabel` en `FrameLabel`. (c) Ajouter `AspectRatio -> 1`.

5.2.3 Utiliser Plot pour estimer les solutions d'une équation

16. Montrer avec Plot qu'il y a 11 solutions à l'équation $2\cos^2(2 * x) - \sin(x/2) = 1$ dans l'intervalle $[0, 3\pi]$. Adapter l'intervalle si nécessaire.

17. Chercher les solutions de $\cosh(\pi x)\cos(\pi x) = -1$. Tester différents intervalles pour x et utiliser après l'option `PlotRange` pour adapter l'intervalle des valeurs en y.

5.3 Manipulate

`Manipulate` permet de créer des graphiques ou interfaces interactives. Sa syntaxe est très similaire à la fonction `Table`, tester ces deux commandes :

```
Table[n, {n, 1, 20}]
Manipulate[n, {n, 1, 20}]
```

Comme avec `Table` on peut mettre n'importe quel fonction en premier argument, comme par exemple `Plot` :

```
Manipulate[Plot[Sin[n*x], {x, 0, 2*Pi}], {n, 1, 20}]
```

On peut aussi créer un Manipulate avec plusieurs paramètres à changer :

```
Manipulate[Plot[Sin[n1*x] + Sin[n2*x], {x, 0, 2*Pi}, PlotRange -> 2],
                                        {n1, 1, 20}, {n2, 1, 20}]
```

18. Étudier avec `Manipulate` le comportement du graphe de $\sin(x) - \cos(nx)$ entre 0 et π quand n change entre 1 à 100.

5.4 ListPlot

Quand on veut tracer des données numériques et non une fonction analytique, on peut utiliser `ListPlot` ou `ListLinePlot` :

```
t = Table[i^2, {i,10}]
ListPlot[t]
ListLinePlot[t]
ListPlot[{t, 2*t}]
```

Si on a une liste de points (x,y), on peut utiliser ListPlot sans modification :

```
t = Table[{i^2, i^3}, {i, 10}]
ListPlot[t]
```

On peut utiliser ListPlot pour illustrer la différence entre = et :=

```
x=Random[];
ListPlot[Table[x,{200}]]
y:=Random[]
ListPlot[Table[y,{200}]]
```

5.5 ParametricPlot

Plot trace y en fonction x. Dans un graphe paramétrique chaque point (x,y) dépend d'un paramètre t par exemple. En regardant sur
tutorial/ParametricPlots
on obtient la syntaxe de ParametricPlot :
`ParametricPlot`[$\{f_x, f_y\}$, $\{t, t_{min}, t_{max}\}$]
Tester un exemple avec $x = \sin(t)$, $y = \sin(2t)$:

```
ParametricPlot[{Sin[t], Sin[2*t]}, {t, 0, 2*Pi}]
```

Dessiner la courbe du "papillon" , découvert par Temple H. Fay :

```
x[t_] := Sin[t] (E^Cos[t] - 2 Cos[4*t] - Sin[t/12]^5)
y[t_] := Cos[t] (E^Cos[t] - 2 Cos[4*t] - Sin[t/12]^5)
ParametricPlot[{x[t], y[t]}, {t, -50, 50}]
```

5.6 ContourPlot

Pour trouver tous les points (x,y) qui satisfont l'équation $x^4 - (x^2 - y^2) = 0$ on peut utiliser `ContourPlot` :

```
ContourPlot[x^4 - (x^2 - y^2) == 0, {x, -1, 1}, {y, -1, 1}]
```

`ContourPlot` ou `DensityPlot` sont utilisées surtout pour afficher des données 3D sur un plan 2D, comme les lignes de hauteur sur une carte. Essayer ces exemples sur la fonction $f(x, y) = \sin(x)\sin(y)$:

```
ContourPlot[Sin[x]*Sin[y], {x, -2, 2}, {y, -2, 2},
        ColorFunction -> ColorData["Rainbow"], PlotLegends -> Automatic]
Plot3D[Sin[x]*Sin[y], {x, -2, 2}, {y, -2, 2},
        ColorFunction -> ColorData["Rainbow"], PlotLegends -> Automatic]
DensityPlot[Sin[x]*Sin[y], {x, -2, 2}, {y, -2, 2},
        ColorFunction -> ColorData["Rainbow"], PlotLegends -> Automatic]
```

5.7 RegionPlot

Pour marquer la région sur le plan 2D où les points (x,y) satisfont une inégalité, comme $x^4 + (x - 2y^2) > 0$, on peut utiliser `RegionPlot` :

```
RegionPlot[x^4 + (x - 2*y^2) >= 0, {x, -2, 2}, {y, -2, 2}]
```

Ou encore :

```
RegionPlot[x^2 + y^2 < 1, {x, -1, 1}, {y, -1, 1}, Mesh -> 10,
        MeshShading -> {{Automatic, None}, {None, Automatic}},
        ColorFunction -> "DarkRainbow"]
```

5.8 PolarPlot

Pour faire des graphes en coordonnées polaires, il existe la fonction `PolarPlot`. Exemple appliqué à $r = 3\cos(6\theta)$:

```
PolarPlot[3*Cos[6*t], {t, 0, 2*Pi}]
```

6 Pour aller plus loin

Ceux qui auront déjà terminé tous les exemples et exercices avant la fin de la séance peuvent étudier quelques “Problem” au choix du livre de Hazrat (voir sur SAKAI sous le dossier “livres”) et d’essayer de résoudre quelques “Exercice” (solutions à la fin du livre).