C. Armana
armana@math.jussieu.fr

## MK1 "Calcul formel" Maple

# TP2 : Séquences, ensembles, listes, fonctions, représentations graphiques

**Accéder à Maple en dehors des heures de TPs :**
Maple est disponible sur les ordinateurs de la salle A, au Script (barre 65-66, deuxième étage). La salle n'est pas tout le temps en accès libre, aussi renseignez-vous sur les horaires auprès du Script. Pour travailler sur les ordinateurs, il vous faudra un compte informatique (amenez la première fois votre carte d'étudiant pour la création du compte).

**Restart au début de chaque exercice :**
Il est très fortement recommandé de mettre une commande :

```
> restart;
```

au début de chaque nouvel exercice.

**Interface de Maple :**
Pour créer une nouvelle ligne d'invite dans Maple, cliquer sur l'icône [> .
Pour taper du texte non interprété par Maple (par exemple, "Exercice n°1"), créer une ligne d'invite avec [> puis cliquer sur l'icône T.
Pour mettre des commentaires au milieu de commandes Maple, on utilise le symbole # (tout ce qui suit le symbole jusqu'à la fin de la ligne est alors ignoré) :

***Et surtout, n'oubliez pas de vous (et de me) poser des questions !***

## 1. Les collections d'expressions

La semaine dernière, nous avons utilisé des variables, en leur affectant une donnée simple (un nombre entier, un nombre complexe,...). Il peut être intéressant de *regrouper plusieurs "objets" dans une seule variable*. Par exemple, si on affecte dans une variable l'ensemble des solutions d'une équation, cela permettra de manipuler toutes les solutions à la fois. Il existe différentes façons de collecter des expressions sous Maple.

### 1.1 Les séquences

Pour Maple, une *séquence* est une collection **ordonnée** d'expressions séparées par des **virgules**. Ces expressions peuvent être du même type (par exemple des nombres entiers) ou des objets de natures différentes (on peut former une séquence composée d'une fonction, d'un nombre entier, et d'un nombre réel approché).

```
> a,b,c ; # exemple de séquence
```

$$a, b, c$$

```
> whattype(%);
```

$$exprseq$$

Le type des séquences est *exprseq*.
Pour former une séquence, **une première méthode** est donc d'en lister les éléments dans l'ordre en les séparant par des virgules. On peut affecter cette séquence à une variable (c'est-à-dire lui "donner un nom") :

```
> s:=a,b,c;
```

$$s := a, b, c$$

Voici sur des exemples comment rappeler les éléments d'une séquence (attention, on utilise des *crochets*) :

```
> s[1]; # premier élément
  s[3]; # troisième élément
  s[4];
```

$$a$$

$$c$$

```
Error, invalid subscript selector
```

Bien sur, il n'existe pas de 4ème élément, c'est pour ça que Maple nous retourne un message d'erreur !

```
> s[2..3];
```

$$b, c$$

La séquence vide (aucun élément) se note NULL :

```
> vide:=NULL;
```

$$vide :=$$

On peut former une séquence en mettant bout à bout (concaténant) plusieurs séquences existantes : il suffit de séparer les séquences par des *virgules* :

```
> t:=1,2,3;
```

$$t := 1, 2, 3$$

```
> u:=t,NULL,s;
```

$$u := 1, 2, 3, a, b, c$$

Remarque : les séquences sont des **structures à lecture seule**Cela signifie qu'on ne peut pas modifier une séquence existante, par exemple en affectant l'un de ses éléments :

```
> u[5];
```

$$b$$

```
> u[5]:=0;
Error, invalid assignment (1, 2, 3, a, b, c)[5] := 0; cannot assign to an
expression sequence
```

Par contre, rien ne nous empêche de créer une nouvelle séquence à partir de "morceaux" de la séquence existante.
**Une deuxième méthode** pour créer une séquence est la fonction *seq*. Elle permet de former seulement les séquences définies à partir de "formules" (voir les exercices).

### 1.2 Les ensembles

Pour Maple, un *ensemble* est une collection **non ordonnée** d'expressions **toutes différentes** . Il se note à l'aide d'une séquence entre **accolades** { , }.
La collection est non ordonnée : l'ordre des éléments que nous donnons à Maple en entrée n'est pas toujours respecté. D'ailleurs, il peut changer d'une fois sur l'autre.

```
> e:={d,b,c,a};
```

$$e := \{a, b, c, d\}$$

Les expressions sont toutes différentes : si Maple voit plusieurs fois apparaitre la même expression, il élimine les doublons :

```
> f:={a,a,a,a,b};
```

$$f := \{a, b\}$$

```
> whattype(e);
```

$$set$$

Le type des ensembles est *set* (ensemble, en anglais). Les méthodes de construction de séquences permettent aussi d'obtenir des ensembles : il suffit d'ajouter des accolades. Par exemple :

```
> {seq(i,i=1..15)};
```

$$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$$

Remarque : comme les séquences, les ensembles sont des **structures à lecture seule.**

On peut effectuer les manipulations usuelles sur les ensembles grâce aux fonctions suivantes :

- la réunion de deux ensembles : *union*
- l'intersection de deux ensembles : *intersect*
- la différence de deux ensembles : *minus*

```
> e minus {a,d};
```

$$\{b, c\}$$

On obtient le nombre d'éléments de l'ensemble par la commande *nops* :

```
> nops(e);
```

$$4$$

et la *séquence* des éléments de l'ensemble par la commande *op* :

```
> op(e);
```

$$a, b, c, d$$

### 1.3 Les listes

Pour Maple, une *liste* est une collection **ordonnée** d'expressions, notée entre **crochets**. C'est une "séquence entre crochets" et il peut y avoir des doublons. On peut la construire à partir d'une séquence.

```
> L:=[seq(1/(2*i),i=1..30)]; whattype(L);
```

$$L := \left[\frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16}, \frac{1}{18}, \frac{1}{20}, \frac{1}{22}, \frac{1}{24}, \frac{1}{26}, \frac{1}{28}, \frac{1}{30}, \frac{1}{32}, \frac{1}{34}, \frac{1}{36}, \frac{1}{38}, \frac{1}{40}, \frac{1}{42}, \frac{1}{44}, \frac{1}{46}, \frac{1}{48}, \frac{1}{50}, \frac{1}{52}, \frac{1}{54}, \frac{1}{56}, \frac{1}{58}, \frac{1}{60}\right]$$

$$list$$

Pour accéder au k-ème élément, on procède comme pour les séquences :

```
> L[4];L[10..15];
```

$$\frac{1}{8}$$

$$\left[\frac{1}{20}, \frac{1}{22}, \frac{1}{24}, \frac{1}{26}, \frac{1}{28}, \frac{1}{30}\right]$$

Remarque :comme les séquences et les ensembles, les listes sont des **structures à lecture seule.**

Pour obtenir le nombre d'éléments de la liste :

```
> nops(L);
```

$$30$$

Pour obtenir la *séquence* formée des éléments de la liste :

```
> op(L);
```

$$\frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16}, \frac{1}{18}, \frac{1}{20}, \frac{1}{22}, \frac{1}{24}, \frac{1}{26}, \frac{1}{28}, \frac{1}{30}, \frac{1}{32}, \frac{1}{34}, \frac{1}{36}, \frac{1}{38}, \frac{1}{40}, \frac{1}{42}, \frac{1}{44}, \frac{1}{46}, \frac{1}{48}, \frac{1}{50}, \frac{1}{52}, \frac{1}{54}, \frac{1}{56}, \frac{1}{58}, \frac{1}{60}$$

On souhaite fondre deux listes en une.

```
> restart;L1:=[a,b,c];L2:=[d,e,f];
```

$$L1 := [a, b, c]$$

$$L2 := [d, e, f]$$

```
> L1,L2;
  whattype(L1,L2);
```

$$[a, b, c], [d, e, f]$$

$$exprseq$$

L'objet que l'on obtient en les séparant par une virule est une séquence de deux listes. Ce n'est pas le résultat voulu. La façon de fondre ces deux listes en une est de concaténer les séquences des éléments de L1 et L2 et de créer une nouvelle liste :

```
> L:=[op(L1),op(L2)];
```

$$L := [a, b, c, d, e, f]$$

## 2. Les fonctions

Nous avons vu que Maple connait des fonctions par défaut : *exp, ln, sin, sqrt, ...* Vous pouvez aussi définir vos propres fonctions. Pour cela, il y a plusieurs méthodes.

### 2.1 L'opérateur flèche ->

Prenons un exemple :

```
> restart; f(x):=x^2; f(3);
```

$$f(x) := x^2$$

$$f(3)$$

Maple ne sait pas calculer f(3), parce que nous ne lui avons pas défini correctement la fonction $f$ !

**Règle importante : sous Maple, on ne définit pas une fonction par f(x):= ...**

L'opérateur flèche -> permet de définir une fonction d'une façon très proche de la notation mathématique. La syntaxe est: nom_fonction : = nom_variable -> expression_en_variable

```
> f:=x->x^2; f(3); f(10);
```

$$f := x \to x^2$$

$$9$$

$$100$$

On peut aussi définir des fonctions de plusieurs variables : nom_fonction : = (séquence_noms_variables) -> expression_en_variables

```
> g:=(a,b)->a^b; g(2,3);
```

$$g := (a, b) \to a^b$$

$$8$$

(attention à bien mettre la séquence des variables entre parenthèses)

Le résultat d'une fonction peut être d'un type quelconque, par exemple une liste :

```
> h:=x->[cos(x),sin(x)]; h(Pi/6);
```

$$h := x \to [\cos(x), \sin(x)]$$

$$\left[\frac{\sqrt{3}}{2}, \frac{1}{2}\right]$$

### 2.2 La fonction *unapply*

Une autre méthode pour définir une fonction est d'utiliser *unapply* dont la syntaxe est :
unapply(expression, séquence_noms_variables).

```
> restart; f:=unapply(x^2,x);f(3);
```

$$f := x \to x^2$$

$$9$$

Pour une fonction de plusieurs variables :

```
> g:=unapply([x^2+y^2,x*y],x,y);
```

$$g := (x, y) \to [x^2 + y^2, x\,y]$$

## 2.3 La composition des fonctions

La composition des applications se fait à l'aide de @.

```
> restart;f:=exp@ln;
```

$$f := \exp@\ln$$

```
> f(x);
```

$$x$$

## 3. Les fonctions add et mul

Elles permettent de calculer des *sommes* et des *produits*. Leur syntaxe est la suivante :
add(f(i),i=a..b) calcule la somme des f(i), pour i entier variant de a à b
mul(f(i),i=a..b) calcule le produit des f(i), pour i entier variant de a à b
Bien sur, i, a et b peuvent avoir d'autres noms. Mais i doit etre une variable non affectée.

```
> add(cos(x)^k,k=0..7);
```

$$1 + \cos(x) + \cos(x)^2 + \cos(x)^3 + \cos(x)^4 + \cos(x)^5 + \cos(x)^6 + \cos(x)^7$$

## 4. La fonction map

La fonction *map* permet, entre autres, d'appliquer une fonction (d'une variable) à tous les éléments d'un ensemble ou d'une liste.

```
> restart; f:=x->x^3;
```

$$f := x \to x^3$$

```
> E:={a,b,c,d,e}; L:=[a,b,d,c,e];
```

$$E := \{a, b, c, d, e\}$$

$$L := [a, b, d, c, e]$$

```
> map(f,E);
```

$$\{a^3, b^3, c^3, d^3, e^3\}$$

```
> map(f,L);
```

$$[a^3, b^3, d^3, c^3, e^3]$$

Attention, on ne peut pas appliquer la fonction *map* à une séquence ! Voici un exemple :

```
> S:=a,b,c,d,e; map(f,S);
```

$$S := a, b, c, d, e$$

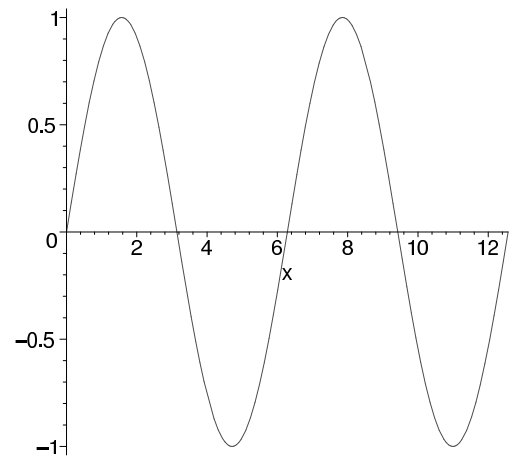
$$a^3$$

On voit que la fonction s'est appliquée uniquement au premier élément de la séquence.

## 5. Les représentations graphiques

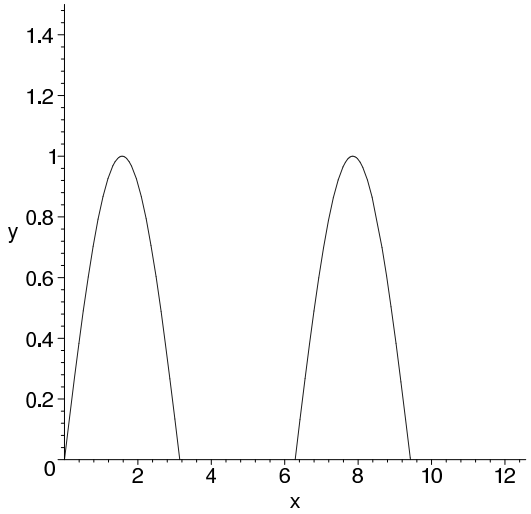
Nous allons voir comment tracer le graphique d'une fonction f définie sur **R** (ou un intervalle de **R**) à valeurs dans**R**. On utilise pour cela la fonction *plot* (qui permet de tracer également d'autres types de graphiques). La syntaxe est la suivante : plot ( f(x) , x=a..b) où a et b désignent les bornes de l'intervalle (de l'axe des abscisses) sur lequel on trace *f*.

```
> restart;
  plot(sin(x),x=0..4*Pi);
```



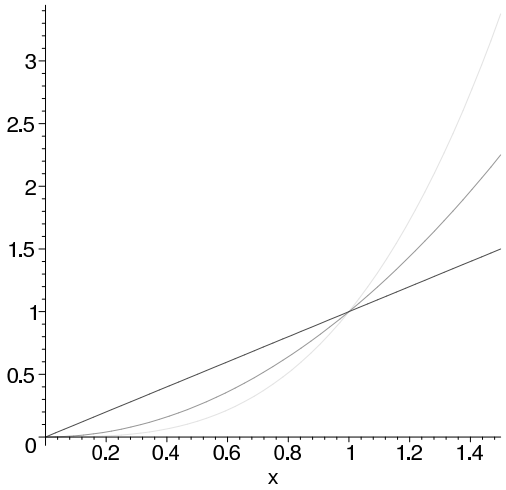

Référez-vous à l'aide de *plot* pour en savoir plus. On peut par exemple spécifier un intervalle en ordonnée pour le tracé et choisir la couleur de la courbe.

```
> plot(sin(x),x=0..4*Pi,y=0..1.5,color=blue);
```




Pour tracer plusieurs fonctions *f1*,...,*fn* à la fois, on utilise la syntaxe *[f1(x),...,fn(x)]* :

```
> plot([x,x^2,x^3],x=0..1.5);
```



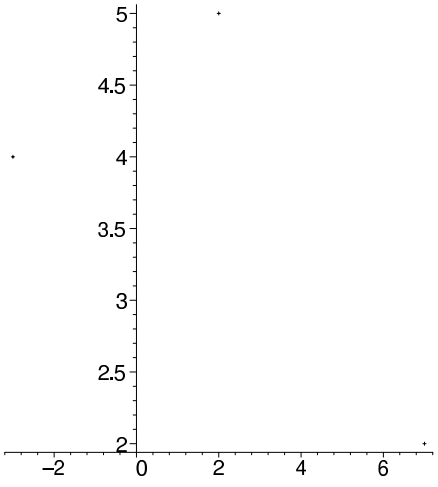

Il existe des fonctions avancées permettant de tracer d'autres types de graphiques. La plupart sont contenues dans la **bibliothèque** (package en anglais) nommée **plots**. Une bibliothèque est un ensemble de commandes et fonctions relatives à un thème (ici, la représentation graphique) qui ne sont pas disponibles par défaut au lancement de Maple. Si vous souhaitez utiliser ces fonctions, vous devez charger la bibliothèque avec la commande *with*.

```
> with(plots);
Warning, the name changecoords has been redefined
```

[*animate*, *animate3d*, *animatecurve*, *arrow*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *conformal3d*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *cylinderplot*, *densityplot*, *display*, *display3d*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *graphplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *interactive*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *odeplot*, *pareto*, *plotcompare*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*, *polyhedra_supported*, *polyhedraplot*, *replot*, *rootlocus*, *semilogplot*, *setoptions*, *setoptions3d*, *spacecurve*, *sparsematrixplot*, *sphereplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*]

Maple affiche la liste des nouvelles fonctions disponibles. Par exemple, pour placer des points dans un plan en donnant leurs coordonnées, on peut utiliser la fonction *pointplot*. Pour avoir de l'aide sur la commande *pointplot*, on doit préciser que la fonction est dans la bibliothèque **plots** :

```
> ?plots,pointplot
> pointplot({[2,5],[-3,4],[7,2]});
```




```
> 
```

**Université Paris 7** **Premier semestre 2005-2006**
**Licence 1ère année** **MK1 - Maple**

## Feuille d'exercices n°2

---

**Exercice 1.** Regarder l'aide à propos de la commande *seq* et comprendre sa syntaxe. L'utiliser pour construire :

- la séquence des $k^2$ pour les entiers $k$ variant de 1 à 30 ;
- la séquence des nombres pairs entre 1 et 50 ;
- la séquence suivante : $1, \frac{1}{8}, \frac{1}{27}, \frac{1}{64}, \frac{1}{125}$ ;
- la séquence des $e^{a\pi/b}$ pour $a$ variant de 1 à 15 et $b$ variant de 1 à 10.

**Exercice 2.** Nous avons déjà vu la commande *solve* pour résoudre des équations. Trouver les racines complexes du polynôme $x^{20} - 1$. Quel est le type d'objet que retourne Maple ? En utilisant Maple et sans les afficher, déterminer combien il a de racines. A l'aide de la fonction *map*, vérifier que ce sont bien des racines. Montrer que la somme des racines est nulle.

**Exercice 3.** Construire en une seule commande :

- la liste des 100 premiers nombres premiers (regarder la commande *ithprime*) ;
- la liste des nombres premiers entre 1 et 100.

**Exercice 4.** Tracer les graphes de $x \mapsto \log(x+1)$ et de $x \mapsto 1,01 \log(x)$. Chercher leur(s) point(s) d'intersection : avec *solve*, que se passe-t-il ? avec *fsolve* ? Tracer sur un même dessin les graphes de ces deux fonctions, en choisissant l'intervalle en abscisse de façon à faire apparaître ce(s) point(s) d'intersection.

**Exercice 5.** Utiliser *seq* pour tracer sur un même dessin les graphes des fonctions $f_n(x) = x^{n^2/10}$ pour $n$ allant de 1 à 10. Recadrer le dessin en abscisse et en ordonnée pour le rendre plus lisible.

**Exercice 6.** On reprend l'exercice 2. En utilisant *pointplot*, tracer dans le plan les points correspondant aux racines complexes du polynôme $x^{20} - 1$. Que remarquez-vous ? Comment l'expliquez-vous ?