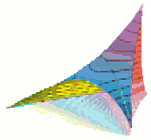

MI 101 Maple
Polynômes - Listes et ensembles
Feuille de TD numéro 3 v. 2 - jeudi 18 octobre

Table des matières

1 Infos diverses

Le site web du cours existe enfin, à l'adresse http://www.mi101.fr.st, et vous y trouverez les documents de cours, les devoirs et examens, les corrigés, ainsi que quelques infos annexes. C'est encore en chantier, mais fonctionnel (mais... en chantier).

2 Polynômes

En maths, un polynôme peut être vu comme un 'objet algébrique', avec ses propriétés intrinsèques, ou par l'intermédiaire d'une fonction polynômiale. On retrouve dans Maple ces deux manières de le construire. Par exemple, pour $7x^2 - (x+1)(x-1)$, on peut

- définir la variable `PE` contenant l'expression `7*x**2-(x+1)*(x-1)`, par `PE:=7*x**2-(x+1)*(x-1)`. Il est imdispensable que `x` soit une variable libre, non affectée. Pour obtenir une valeur particulière (avec $x = -1$ par exemple), on peut affecter la valeur souhaitée à la variable `x`, ou remplacer `x` par la valeur souhaitée dans la variable `P`, par la substitution `subs(x=-1,PE)`.
- définie la fonction polynômiale $PF(t)$, par `PF:=(t)->7*t**2-(t+1)*(t-1)`. Ici, l'utilisation de `t` plutôt que `x` ou `u` est quelconque; il s'agit de la variable de la fonction, son nom est sans importance et n'a aucun lien avec les variables Maple créées ou utilisées dans d'autres commandes. Pour obtenir une valeur particulière (avec $t = -1$ par exemple), il suffit de saisir l'instruction `PF(-1)`.

Ces deux définititions sont 'interchangeables'.

- On obtient `PF` à partir de `PE` par `PF:=(x)->PE` ou `PF:=unapply(PE,x)`.
- On obtient `PE` à partir de `PF` par `PE:=PF('x')`.

ATTENTION : les fonctions Maple détaillées ci-après traitent des expressions polynômiales, pas des fonction.

Une fois défini le polynôme, on peut le manipuler, le développer, le factoriser, l'ordonner, extraire les racines,...

- `simplify()`, déjà vu, effectue les simplifications élémentaires de toute expression, donc entre autres de polynômes.
- `factor()` factorise le polynôme. Il est possible de spécifier par quelle expression on souhaite factoriser.
- `expand()` développe une expression, la simplifie et l'ordonne.
- `sort()` ordonne le polynôme (développé) par ordre décroissant d'exposants.
- `solve()` fournit la liste des racines du polynôme.
- `degree()` nous renvoie le degré du polynôme (son plus grand exposant).

3 Listes et ensembles

On a déjà vu, la semaine dernière, l'utilisation des variables sous Maple, mais uniquement en créant des variables contenant une donnée simple (une valeur numérique, une expression, etc...). Il peut être intéressant de regrouper plusieurs 'objets' dans une seule variable. Par exemple, l'ensemble des nombres premiers entre 1 et 100, ou les racines nièmes de l'unité, ou les valeurs de $\cos(\frac{2k\Pi}{n})$, pour $k \in [0; n-1]$.

Ces regroupements d'objets se font dans des structures appelées séquences, listes et ensembles, de caractéristiques différentes (il existe aussi la structure tableau, proche de la liste, que nous verrons peut-être plus tard).

3.1 Séquences

La manière la plus simple de définir une séquence est la description de ce que Maple nous présente : "une succession finie d'objets séparés par des virgules". En fait, il s'agit de "ce qu'il y a dans les ensembles et les listes". Elles peuvent être utilisées et traitées par Maple dans certaines fonctions, mais sont rarement manipulées en tant que telles. Par contre, Maple fournit souvent ses résultats sous forme de séquence.

Remarque : un objet Maple simple est déja une séquence, contenant un seul élément.

- On crée une séquence à l'aide de la fonction `seq()`, en indiquant entre les parenthèses l'expression générique des objets que contient la séquence, et les valeurs que prennent les indices, comme par `S:=seq(x**2,x=1..10)`.
- Les séquences peuvent être ensuite modifiées : on les concatène ('assemble') en les écrivant côte-à-côte, simplement séparées par une virgule.

3.2 Listes

Une liste est un objet Maple qui contient une séquence. Les éléments de la liste (donc de la séquence) sont ordonnés (on peut obtenir le premier, le second,...) et peuvent se répéter (une liste peut contenir `1,2,1,2,1,2`, et elle contient alors 6 éléments).

- Une liste est construite comme une séquence encadrée par des crochets `[` et `]`.

- Le nombre d'éléments que contient la liste `L` (la longueur de la séquence que contient la liste, en fait) est `nops(L)`.
- La séquence que contient la liste `L` est `op(L)`; le troisième élément de la liste est `op(3,L)`.
- `member(x,L)` teste si la liste `L` contient l'élément `x`. `member()` permet aussi d'obtenir la position d'un élément dans une liste.

Si on souhaite construire la liste `L` contenant les éléments des suites `L1` et `L2`, à la suite, on procède progressivement : il faut extraire les séquences de `L1` et `L2`, les concaténer, et les remettre dans une liste; cela donne `L:=[op(L1),op(L2)]`. ATTENTION : il faut bien voir quel objet on souhaite construire, et ce qu'il doit contenir; par exemple, `L:=[L1,L2]` définit une liste `L` qui contient deux éléments, ceux-ci étant chacun une liste, et `L:=op(L1),op(L2)` définit une séquence qui est la concaténation des séquences de `L1` et `L2`. Bon...c'est peut-être un peu confus, mais je pense que la pratique, associée à vos questions (judicieuses) éclaireront tout cela.

3.3 Ensembles

Un ensemble est un objet Maple qui contient une séquence. A la différence d'une liste, et pour correspondre à la définition mathématique, les éléments de l'ensemble (donc de la séquence) ne sont pas ordonnés (Maple peut modifier l'ordre de stockage en mémoire et d'affichage pour optimiser le traitement d'un ensemble) et ne peuvent se répéter (un ensemble qui contient, au moment de sa définition, `1,2,1,2,1,2`, ne contient en fait que deux éléments, `1` et `2`).

- Un ensemble est construit comme une séquence encadrée par des accolades { et }.
- Le nombre d'éléments que contient l'ensemble `E` (la longueur de la séquence qu'il contient, en fait) est `nops(E)`.
- La séquence que contient l'ensemble `E` est `op(E)`; cependant, à la différence des listes, demander le premier ou le troisième élément d'un ensemble n'a pas vraiment de sens.
- Comme pour les listes `member()` teste l'appartenance d'un élément à un ensemble, et peut donner sa position.
- On peut aussi effectuer des opérations ensemblistes classiques, avec les fonctions `union`, `intersect` et `minus` (attention, syntaxe particulière).

De même que pour les listes, l'ensemble `E` contenant les éléments de `E1` et `E2` est défini par `E:={op(E1),op(E2)}`. Ici, il s'agit en fait de la simple réunion de deux ensembles;

4 Exercices

Ils sont peut-être un peu compliqués, n'hésitez pas à demander des explications ou conseils. Et les thèmes polynômes/listes/ensembles ne sont pas hermétiques.

4.1 Polynômes

Exo 1 Simplifiez, développez, ordonnez, donnez le degré de $((x+3)^3-1)^2-(x^6-1)-2x(3x^2+2)^2$

Exo 2 Retrouvez les expressions des racines des polynômes ax^2+bx+c et ax^3+bx^2+cx+d (on paut aussi faire pour le degré 4, mais on ne sait mathématiquement pas faire au-delà...). Pour le dernier résultat (degré 3), vérifiez avec l'une des racines trouvées que celle-ci annue bien le polynôme. Il peut être utile d'utiliser des commandes de listes ou ensembles.

Exo 3 A partir de monômes $(x-\alpha)$, construisez un polynôme P nul en -2, -1, 1 et 2. Puis modifiez-le pour que $P(0)=1$, tout en conservant les propriétés précédentes. Pour info, ce principe est utilisé pour approcher une fonction par des polynômes...

4.2 Séquences

Exo 4 Affichez la séquence des n^2, pour n entre 1 et 30, et la séquence des nombres pairs entre 1 et 50

Exo 5 b étant une variable non affectée, construisez la séquence des a^b, pour a entre 1 et 10. Construisez ensuite la liste des a^b, pour a ET b entre 1 et 10.

4.3 Ensembles

Exo 6 Vérifiez par un exemple que l'ordre n'a pas de signification dans un ensemble, et qu'il n'y a pas de répétition des éléments.

Exo 7 Simplifiez le polynôme $(z^{10}+1)(z^{10}-1)$. Puis, sans les afficher, déterminez combien il a de racines.

Exo 8 Construisez les ensembles des multiples de 2, 3,..., 10 (à l'exception de 2, 3,..., eux-mêmes) compris entre 1 et 100, faites-en la réunion, puis obtenez-en la liste des nombres premiers entre 1 et 100 (les nombres premiers entre 1 et 100 sont les nombres qui ne sont pas multiples de 2, 3,..., 10; il s'agit du principe du 'crible d'Erathostène').

Exo 9 A partir de valeurs de i et j de votre choix, construisez les ensembles Ei et Ej des racines iièmes et jièmes de l'unité (solutions de $x^i = 1$ et $x^j = 1$). Vérifiez que Ek, défini comme l'intersection des deux ensembles que vous venez de définir, est aussi l'ensemble des racines kièmes de l'unité, k étant le PGCD de i et j.

4.4 Listes

Exo 10 Vérifiez par un exemple que l'ordre est conservé dans une liste, et qu'il peut y avoir répétition des éléments.

Exo 11 Construisez, en une seule commande, la liste des 100 premiers nombres premiers, puis la liste des nombres premiers entre 1 et 100.

Exo 12 Construisez, à partir des résultats de l'exercice prcédent, une liste dont chaque élément est un ensemble, le nième contenant les nombres premiers entre 1 et n, pour n variant de 1 à 100, puis obtenez à partir de cette liste une seconde liste dont le nième élément est le nombre de nombres premiers entre 1 et n (pas clair? demandez-moi des explications...).

5 Corrections

RAPPEL : `restart` entre chaque exercice...

5.1 Polynômes

Exo 1

```
>P:=((x+3)**3-1)**2-(x**6-1)-2*x*(3*x**2+2)**2;
>simplify(P);expand(P);sort(P);sort(expand(P));degree(P);degree(expand(P));
```

Exo 2

```
>P2:=a*x**2+b*x+c;P3:=a*x**3+b*x**2+c*x+d;R2:=[solve(P2,x)];R3:=[solve(P3,x)];
```

Puis

```
>r:=op(1,R3);
>PR:=subs(x=r,P3);simplify(PR);
```

ou

```
>PF3:=unapply(P3,x);simplify(PF3(r));
```

Exo 3

```
>P:=(x+2)*(x+1)*(x-1)*(x-2);Paj:=P/subs(x=0,P);seq(subs(x=i,Paj),i=-2..2);
```

5.2 Séquences

Exo 4

```
>seq(n**2,n=1..30);seq(2*n,n=1..50/2);
```

Exo 5

```
>seq(a**b,a=1..10);
>seq(seq(a**b,a=1..10),b=1..10);
```

5.3 Ensembles

Exo 6

```
>E:={1,3,2,3};
```

Exo 7

```
>P:=(x**10+1)*(x**10-1);P:=simplify(P);nops({solve(P)});
```

Exo 8

```
>M:={seq({seq(i*n,n=1..100/i)},i=2..10)};
```

On a obtenu un ensemble d'ensembles, et on va à présent réunir ces derniers:

```
>MM:={seq(op(op(i,M)),i=1..nops(M));}
```

On ôte enfin `MM` de l'ensemble des nombres entiers positifs inférieurs à 100:

```
NP100:={seq(n,n=2..100)} minus MM;
```

Exo 9

```
>i:=14:j:=12:EI:={solve(x**i-1)};EJ:={solve(x**j-1)};EIJ:=EI intersect EJ;
```

On a construit Ei et Ej, et leur intersection.

```
>k:=igcd(i,j):EK:={solve(x**k-1)};
```

Les deux ensembles sont identiques.

5.4 Listes

Exo 10

```
>E:=[1,3,2,3];
```

Exo 11

```
>NP1:=[seq(ithprime(i),i=1..100)];
>NP2:={seq(nextprime(i),i=0..99)} intersect {seq(i,i=1..100)};NP2:=[NP2];
```

Exo 12

```
>NP:=[seq({op(NP2)} intersect {seq(i,i=1..n)},n=1..100)];
>NB:=[seq(nops(op(i,NP)),i=1..nops(NP))];
```