Université de Bordeaux
Mathématiques

Algèbre et calcul formel – Agrégation
2017–2018

FEUILLE D'EXERCICES n° 2
Équations, polynômes

1. Équations, dérivation, intégration

Pour résoudre une équation, ou un système d'équations, on peut utiliser la commande `solve`.

Attention ! Pour cette fonction, comme pour la plupart des fonctions de Sage, il faut d'abord définir les variables.

1) Essayer la commande
`solve([x-1==0],x)`
Ok, ça marche. Maintenant essayer
`solve([t-1==0],t)`
La variable `x` est une exception. Toute autre variable doit absolument être déclarée au préalable. `t=var('t')`
`solve([t-1==0],t)`
`parent(t)`
La variable ainsi définie appartient donc à l'ensemble `Symbolic Ring`, qu'on peut noter `SR`.
Si on fait `parent(x)`, on voit que la variable `x` est par défaut dans `Symbolic Ring`.
Taper
`reset('t')`
puis
`parent(t)`
On a effacé la variable `t`.

2) Dans `Help`, `Work trough the tutorial`, `Basic algebra and calculus`, faire les exercices des paragraphes *Solving equations exactly*, *Solving equations numerically* puis *Differentiation, integration etc.*

2. Polynômes

Dans sage, toutes les valeurs sont définies en temps qu'éléments d'un ensemble bien précis, que l'on retrouve grâce à la commande `parent`, ou bien `type`. Nous avons vu l'exemple des entiers, des listes, des nombres rationnels et des réels.

Pour d'autres structures, il faut définir au préalable l'ensemble qu'on va considérer. C'est le cas par exemple pour les polynômes.

1) Tester

`p==t^2-2; p`

Ca ne marche pas : il faut définir `t`. Comme ci-dessus, si l'on met `x` plutôt que `t`, cela va fonctionner, mais p ne sera pas vraiment considéré comme un polynôme en `x` et les fonctions adaptées aux polynômes ne fonctionneront pas de la même manière. En effet, `x` est par défaut un élément de `Symbolic Rings`.

2) On va définir l'anneau de polynômes $\mathbb{Q}[x]$,et explorer quelques fonctions définie sur cet anneau.

Expérimenter les commandes suivantes.

`QQ` (C'est le corps $\mathbb{Q}$ des nombres rationnels)

`Qx.<x>=PolynomialRing(QQ)`

Cette commande définit `Qx` comme étant l'anneau de polynômes $\mathbb{Q}[x]$ et la variable `x` devient la variable x de cet anneau.

`Qx`

`parent(x)`

`p=x^2-2; p`

`p=Qx([-2,0,1]); p` (c'est une autre façon de définir le même polynôme).

`p.degree()`

`p.is_irreducible()`

`factor(p)`

`p.roots()`

`p.roots?` (pour demander le mode d'emploi de la fonction)

`p.roots(ring=RealField())`

`p(2); p[2]`

`p.coefficients()`

`p.coefficients(sparse=false)`

On peut aussi définir $\mathbb{R}[x]$:

`Rx.<x>=PolynomialRing(RR)`

Si on veut considérer p comme un polynôme de $\mathbb{R}[x]$, on fait :

`p=Rx(p)`

Alors, la commande `p.roots()` donnera les racines de `p` dans son corps de base, c'est-à-dire $\mathbb{R}$. Nous verrons plus tard d'autres corps ou anneaux que $\mathbb{Q}$ et $\mathbb{R}$.

Remarque. Plus généralement, si l'on a défini un objet A, qui est un certain ensemble ou structure, et si a est un objet susceptible de définir un élément de A, alors on définit cet élément par la commande $A(a)$.

Par exemple, si $A = \mathbb{Z}/n\mathbb{Z}$ (nous verrons plus tard comment définir cet objet) et si a est un entier, alors $A(a)$ est l'élément de $\mathbb{Z}/n\mathbb{Z}$ défini par a.

Nous avions déjà utilisé cette construction, par exemple quand on a posé ci-dessus

`p=Qx([-2,0,1])`, nous avons défini un élément de `Qx` (c'est-à-dire $\mathbb{Q}[x]$) à partir d'une liste. De même, `list(p)` permettrait de définir une liste à partir du polynôme `p`.

Nous retrouverons ce principe à maintes reprises.

3) Il y a d'autres façons de définir les anneaux de polynômes. Dans `Help`, `Work trough the tutorial`, faire les exercices de `Polynomials`.

4) Expérimenter les commandes suivantes

```
PR=PolynomialRing(QQ,3,'a');PR
p=a0+2*a1
```

Problème ! Il faut donner un nom aux variables.

```
v=PR.gens();v
p=v[0]+2*v[1];p
```

5) Nous allons ici calculer toutes les solutions polynomiales de degré inférieur ou égal à 2 de l'équation différentielle

$$(3t-2)y'' + 2t^2y' - 2(2t-1)y = 0 \tag{1}$$

en utilisant un polynôme à coefficients indéterminés.

Nous allons ici utiliser la commande `solve`. Lorsque nous aurons vu (ou revu) le calcul matriciel sur Sage, nous pourrons refaire le même exercice en utilisant les matrices.

Il y a différentes manières de faire. Par exemple :

`SRt.<t>=PolynomialRing(SR)` (c'est l'anneau des polynômes de variable `t` sur l'anneau `Symbolic Ring`)

```
a0,a1,a2 = var('a0 a1 a2')
p = a0+a1*t+a2*t^2
p1 = diff(p,t)
p2 = diff(p1,t)
q = (3*t-2)*p2+2*t^2*p1-2*(2*t-1)*p
l = q.coeffs()
solve(l,[a0,a1,a2])}
```

C'est une possibilité, mais comment faire si l'on veut automatiser ce calcul en une fonction qui prendrait le degré maximal du polynôme recherché en paramètre ? Cela ne semble pas évident...

Voici une autre manière de faire, qui semble plus facile à automatiser. Commençons par définir les indéterminées. Pour cela, on définit un anneau de polynômes à plusieurs variables (ces variables seront les variables de notre polynôme en t).

```
PR=PolynomialRing(QQ,3,'b');PR
```

`PR` est l'anneau des polynômes à coefficients dans $\mathbb{Q}$ et à trois indéterminées `b0, b1, b2`. Si on tape `b0`, on voit qu'il y a un problème, comme on l'a déjà vu dans la question **4**. Il faut renommer les variables.

```
v=PR.gens();v;v[1]
```

C'est mieux, mais les listes sont plus maniables que les n-uplets, c'est pourquoi je préfère écrire

```
v=list(PR.gens());v
```

La commande `list('***')` transforme `'***'` en liste, conformément à la remarque générale de la fin du **2.**

Reste à définir l'anneau des polynômes en t.

```
PRt.<t>=PolynomialRing(PR);PRt
```

`PRt` est donc l'anneau $\mathbb{Q}[b_0, b_1, b_2][t]$, et chaque indéterminée b_i correspond ici à `v[i]`. On cherche les polynômes de la forme

$$P = b_2 t^2 + b_1 t + b_0$$

qui sont solutions de (1). On pose

```
p=v[2]*t**2+v[1]*t+v[0]
```

Puis on fait comme ci-dessus

```
p1 = diff(p,t);
p2 = diff(p1,t)
q = (3*t-2)*p2+2*t^2*p1-2*(2*t-1)*p
l = q.coeffs()
solve(l,v)
```

Ah ! Cela ne marche pas ! C'est que dans `solve`, il faut que les `v[i]` soient des éléments de `Symbolic Ring`.

Pour cela, on peut faire :

```
for i in range(3):
  v[i]=SR(v[i])
```

```
for i in range(3):
  l[i]=SR(l[i])
```

et ensuite seulement :

```
solve(l,v)
```

(c'est sur la boucle `for` ci-dessus que le fait que `v` soit une liste plutôt qu'un triplet est utile : si `v` est un triplet, on ne peut pas modifier ses composantes).

Cette dernière commande affiche alors les coefficients des solutions de l'équation. Cela peut nous suffire, mais on peut aussi affiner en affichant les polynômes solutions eux-mêmes. Pour cela, on peut utiliser la commande

```
S=solve(l,v,solution_dict=True)
```

Alors, `S` est la liste des solutions. Ici, il n'y en a qu'une : `S[0]`, qui est un dictionnaire dont les entrées sont numérotées par les `v[i]`. Ainsi, les solutions sont données par

```
sum([S[0][v[i]]*t**i for i in range(3)])
```

6) Écrire une fonction qui, étant donnés a, b et c dans $\mathbb{Q}[t]$ et un entier d, donne les polynômes de degré inférieur ou égal à d solutions de l'équation

$$ay'' + by' + cy = 0.$$