

## COMPÉTENCES ACQUISES EN SÉANCE 1

- Prendre ses repères dans l'aide. (Tutoriel, aide sur mots clefs, menus, ...)
- Différences symboles/fonctions. Ex : si la variable R contient le symbole  $(a+b)/(1-a^2)$  comment créer les fonctions  $a \mapsto \frac{a+b}{1-a^2}$  et  $(a,b) \mapsto \frac{a+b}{1-a^2}$  à partir de R? (cf. `unapply`)
- Libérer une variable pour pouvoir l'utiliser comme symbole. (cf. `purge` ou `del`)
- Premières matrices.
- Quelques dessins.

## MODES XCAS ET PYTHON

- ⚠ En 2019 de nombreuses versions d'Xcas ont des problèmes lorsque la configuration par défaut est en mode Python. Il faut donc **choisir le mode Xcas lors du premier lancement**, même si l'on souhaite utiliser une syntaxe comme en python. Si au lancement vous êtes en mode python, allez dans le menu `Cfg>config` du `Cas` changez le mode vers `xcas`, **sauvez** la config et **quittez** Xcas puis relancez le.
- Pour utiliser un symbole dans une fonction (par exemple `X`) il vaut mieux<sup>2</sup> faire

(giac/xcas)

```
local X;
purge(X)
```

- Dans Xcas, commencer une cellule par le symbole `#` bascule son contenu en mode python quelle que soit la configuration globale (et commencer par `//` bascule en mode Xcas).

(giac/xcas)

```
# La cellule sera en mode python, ceci donnera donc 0 (le quotient de 3 par 4)
3//4
A:=5 # on peut alors utiliser l'affectation comme en python. NB := marche aussi.
```

(giac/xcas)

```
// La cellule sera en mode Xcas, ceci donnera donc 3 suivi du commentaire 4
3//4
A:=5; // ici l'affectation doit se faire avec :=
```

- ⚠ **Mais attention** à ne pas valider une cellule qui ne contient que `#` car cela peut planter<sup>3</sup> Xcas immédiatement.
- NB : Dans une fonction définie par `def` on est automatiquement en mode python. L'exemple ci dessous donnera toujours 0.

(giac/xcas)

```
def test():
    return 3//4
```

## Exercice I: listes, modulaires

Dans `xcas` (à partir de 1.4.9) et `sage`, on peut créer des listes automatiquement avec une syntaxe type python suivante :

```
[ 1/k for k in range(1,11) ]
```

1. <http://www.imj-prg.fr/~han/agreg>
2. certaines "anciennes" versions déclarerait automatiquement toutes les variables comme locales dans une fonction python, et leur affecterait 0, d'où le besoin de faire `purge/del` pour avoir un symbole.
3. semble OK avec la vieille version 1.4.9 sur `clefagreg 2019`

dans xcas on pourrait aussi faire de manière équivalente : `[seq(1/k, k=1..10)]` mais attention, cette fonction a de nombreuses options qui retournent des choses différentes<sup>4</sup> pour des raisons de compatibilité avec d'autres syntaxes. Si vous l'utilisez tester toujours votre façon de l'utiliser.

- 1) Créez une liste L1 des éléments non nuls de  $\mathbb{Z}/17\mathbb{Z}$ .
- 2) Créez la liste des inverse des éléments de L1
- 3) En créant un ensemble, donnez la liste des éléments du groupe engendré par 2 dans  $(\mathbb{Z}/17\mathbb{Z})^\times$ .

### Exercice II: (Petits programmes)

Dans le tutoriel de votre logiciel vous trouverez les syntaxes usuelles de programmation. Sous Xcas dans une entrée normale on fait Shift+Return pour aller à la ligne sans évaluer. Mais pour un programme plus long, il est plus confortable d'insérer un éditeur de programme via le menu : `Programme>Nouveau programme`. Sous Sage, on fait Shift+Return pour évaluer et Return pour aller à la ligne sans évaluer

- 1) Faire l'exercice V de la séance 1.
- 2) Définir une fonction  $g : x \mapsto \sin(x) - x/2$ , dessinez son graphe et créez un programme `dicho(f, a, b, epsi)` qui cherche une racine d'une fonction  $f$  entre  $a$  et  $b$  avec  $f(a)f(b) < 0$  à la précision `epsi`.

**Fonction** `dicho(f, a, b, eps)`

**tantque**  $b - a > eps$  **faire**

$c \leftarrow \frac{a + b}{2};$

**si**  $f(a)f(c) \leq 0$  **alors**

$b \leftarrow c;$

**sinon**

$a \leftarrow c;$

**fsi**

**ftantque**

**retourner**  $a, b;$

---

*a.* mettre une inégalité large, au cas où  $c$  soit une racine de  $f$ .

### Exercice III: Algèbre linéaire

1) a) Dans xcas les vecteurs<sup>5</sup> n'ont pas un type particulier ce sont juste des listes et se notent entre `[ ]`. Par exemple, essayez : `v1:= [j for j in range(1,5)]` et `v2:= [a,b,c,d]` et `t*v1+v2` Dans Xcas, les matrices se notent `M:= [[1,2,3,4], [5,6,7,8]]` ou bien aussi `matrix([[1,2,3,4], [5,6,7,8]])` comme dans Sage. Leur lignes sont `M[0]` `M[1]` ... et les coefficients `M[i, j]`. Dans Xcas si l'on souhaite des indices commençant à 1 au lieu de 0 on utiliser des `M(1)` au lieu de `M[0]`

Les vecteurs s'affichent en ligne, mais ce sont bien des vecteurs et non des matrices lignes. Remarquez la différence d'affichage entre `[v1]` et `v1`, et copiez<sup>6</sup> collez la sortie (noire) de `[v1]` dans une zone d'entrée pour étudier les `[]`.

b) Les lois usuelles en algèbre linéaire utilisent encore `+` et `*`. Calculez les images de  $v_1$  et  $v_2$  par  $M$  (remarquez que la sortie est bien un vecteur). Calculez  $M_2 = M \cdot ({}^t M)$ ,  $M_2^2$ . Comment obtenir rapidement  $M_2^n$  ?

c) Que fait `v1*v2` ?

d) Remarquez que dans xcas la fonction opérande : `op` permet d'enlever un niveau de crochets. On ajoute donc une ligne très facilement sans connaître les instructions dédiées.

2) Une syntaxe très utile dans Xcas et Sage qu'il faut retenir, est de créer une matrice à partir d'une fonction<sup>7</sup> de deux variables en utilisant `matrix` :

a) Par exemple, créez dans votre logiciel la fonction :  $f : (x, y) \mapsto x + 10^y$  puis la matrice suivante :

`matrix(6,6,f);`

b) On souhaite travailler avec un nombre de variables que l'on peut facilement modifier. On aimerait donc des noms du type  $x_1, x_2, \dots$ . Dans sage on peut définir un nouvel anneau de polynômes et injecter

---

4. Ex : réponse entre crochets ou pas

5. dans sage, type : vector

6. Notez que l'on peut copier une sortie pour en faire une entrée, même si l'aspect semble différent. Ça n'est pas toujours facile de sélectionner une sortie à cause de la structure d'arbre d'une formule. Control a selectionne tout

7. Attention, dans sage, il faut un programme et non une fonction symbolique

les variables si besoin. Sous Xcas, une première solution est d'utiliser plutôt des noms du type  $x[j]$  où  $x$  est une lettre inutilisée. On peut alors créer la matrice de Vandermonde de taille 6 ainsi :

(giac/xcas)

```
f1(u,v):=x[u]^v;V:=matrix(6,6,f1);
```

(sage)

```
sage: n=6; R=PolynomialRing(QQ,n,x)
sage: matrix(3,3, lambda a,b: R.gen(a)^b)
[ 1 x0 x0^2]
[ 1 x1 x1^2]
[ 1 x2 x2^2]
```

c) Calculez le déterminant de Vandermonde formel en taille 6x6 et factorisez le.

d) Lorsqu'il y a beaucoup de variables et que la taille de la matrice n'est pas trop grande, le calcul du déterminant par la méthode des mineurs s'avère plus efficace (Je ne sais pas si c'est possible sous Sage). En déduire un moyen d'obtenir le déterminant de Vandermonde en taille 8x8, puis factorisez le. (pour les calculs longs on a intérêt à stocker d'abord le déterminant puis tenter de factoriser.)

*Pour la syntaxe de programmation, regardez le tutoriel dans le menu : Aide/Débuter en Calcul Formel*

#### Exercice IV: blocs, polynômes, matrices nilpotentes

1) (Matrices par blocs)

a) Créez une fonction  $JJ(n)$  qui retourne une matrice de taille  $n \times n$  ayant pour seuls termes non nuls des 1 sur la droite  $j - i = 1$ .

b) MINI PROGRAMME. (Sous xcas on pourra utiliser/découvrir l'éditeur de programme, et regarder le tutoriel).

Créez une fonction  $J(L)$  qui associe à une liste décroissante d'entiers naturels non nuls  $L^8$ , une matrice diagonale par blocs constitués des  $JJ(n)$  correspondants. On cherchera une instruction pour construire des matrices diagonales par blocs, et l'on vérifiera que la suite est bien décroissante.

c) Etudiez sur divers exemples les rangs et noyaux des puissances de  $J(L)$  en fonction de  $L$ . Conclusion?

2) (Matrices et polynômes) Cette année, nous aurons souvent besoin de trouver des matrices d'applications définies par des polynômes.

a) Créez une fonction  $M(P,Q)$  qui aux deux polynômes  $P,Q$  en la variable  $x$  associe la matrice dans la base canonique  $(1, x, \dots, x^{\deg(Q)-1})$  de l'application :

$$\begin{array}{ccc} \mathbb{Q}[x]/(Q) & \longrightarrow & \mathbb{Q}[x]/(Q) \\ R & \mapsto & R \cdot P \end{array}$$

Indications : on cherchera une fonction qui donne à  $i$  fixé le coefficient de  $x^i$  dans un polynôme en  $x$  plutôt que de convertir des polynômes en liste. (Pourquoi? quelle serait la longueur de la liste pour  $1 + x^3, 2 + x, \dots$ ?)

b) Vérifiez que  $M(x + 2, x^4 - 1)$  vaut :

$$\begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 2 \end{pmatrix}$$

3) On considère la matrice  $N = M(P,Q)$  où :

$$P = x^5 - 2 \cdot x^4 + x^3 - 2 \cdot x^2 - 2 \cdot x + 4$$

$$Q = (x^2 - 1)^2 (x^2 + 2)^5 (x^3 - 5 \cdot x^2 + 8 \cdot x - 4)$$

---

8. exemple : [2,2,1,1]

a) Quel est le polynôme caractéristique de  $N$ ? (NB : l'instruction directe est plus efficace que  $\det(A - x.I)$ . Quel est le polynôme minimal de  $N$ ?

b) Comment obtient t'on une base du noyau de  $N$ ? Par exemple, comment calculer la somme du second et troisième vecteur de cette base? Comment obtenir la dimension du noyau de  $N$ ? (On cherchera une technique qui fonctionne aussi pour les applications injectives)

c) En demandant au logiciel de calculer des rangs de certaines matrices, déterminer l'unique  $L$  pour que  $J(L)$  et  $N$  puissent être semblable. (On rappelle que l'on impose à  $L$  d'être décroissante à valeur dans  $\mathbb{N}^*$ , et on n'utilisera pas de commande de Jordanisation du logiciel)

d) Donnez une base de  $\ker N^4$ , et choisissez deux vecteurs  $a_1, a_2$  de  $\ker N^5$  indépendants modulo  $\ker N^4$ . Que devez vous demander au logiciel pour prouver que  $a_1, a_2$  conviennent?

e) Prenez un vecteur  $a_3$  dans  $\ker N^3$  tel que  $N^2(a_1), N^2(a_2), a_3$  soient indépendants modulo  $\ker N^2$  (Prenez en un au hasard et prouvez avec votre logiciel que votre choix de  $a_3$  est légitime).

f) Prenez deux vecteurs  $a_4, a_5$  de  $\ker(N^2)$  tels que  $N^3(a_1), N^3(a_2), N(a_3), a_4, a_5$  soient libres modulo  $\ker(N)$ .

g) On considère la famille de vecteurs

$$N^4(a_1), \dots, N(a_1), a_1, N^4(a_2), \dots, N(a_2), a_2, N^2(a_3), \dots, a_3, N(a_4), a_4, N(a_5), a_5.$$

Vérifiez<sup>9</sup> que c'est une base de  $\mathbb{Q}^{17}$ . Donnez ma matrice  $P_N$  de passage associée, et conjuguez  $N$  pour obtenir une matrice de type J(L).

---

9. vu que l'on a fait les vérifications précédentes, on pourrait démontrer que cette dernière vérification est inutile