Université Paris 8 Année 2021–2022

Licence Mathématiques

Logiciel de calcul formel — Interrogation écrite — Sujet A

11/04/2022

Nom :

Prénom :

Numéro d'étudiant :

Aucun document et **aucun dispositif électronique** n'est autorisé.

Durée : 1h30

Le sujet est composé de 8 exercices indépendants, avec un total de 12 questions.

Vous trouverez ci-dessous une liste de commandes qui pourraient vous être utiles dans la résolution des exercices.

commande	description
`M.nrows()`	retourne le nombre de lignes de `M`
`M.ncols()`	retourne le nombre de colonnes de `M`
`M.derterminant()`	retourne le déterminant de `M`
`M.inverse()`	retourne l'inverse de `M`
`M.right_kernel()`	retourne le noyau à droite de `M`, en tant qu'espace vectoriel
`M.left_kernel()`	retourne le noyau à gauche de `M`, en tant qu'espace vectoriel
`M.solve_right(b)`	retourne un vecteur `x` solution de `M * x = b`
`M.solve_left(b)`	retourne un vecteur `x` solution de `x * M = b`
`V.basis()`	retourne une base de `V`
`V.dimension()`	retourne la dimension de `V`

Table 1 – Quelques fonctions sur les **matrices et espaces vectoriels**. On suppose que `M` est une matrice carrée, que `V` est un espace vectoriel, et que `b` est un vecteur.

commande	description
`randint(a, b)`	retourne un nombre entier tiré uniformément dans l'ensemble $\{a, a+1, \ldots, b\}$.
`ranrange(a, b, r)`	retourne un nombre entier tiré uniformément dans l'ensemble $\{a, a+r, a+2r, \ldots, a+mr\}$ où m est tel que $a+mr < b \leq a+(m+1)r$.
`uniform(u, v)`	retourne un nombre réel tiré uniformément dans l'intervalle $[u, v[$.

Table 2 – Quelques fonctions de **tirage aléatoire**. On suppose que `a`, `b` et `r` sont des entiers, et que `u` et `v` sont des réels.

Exercice 1. Fonctions et limites.

Question 1.– Déclarer une expression symbolique `f` en la variable `x` qui vaut $f(x) = \frac{e^x - 1 - x}{x^2}$.

```
var('x')
f = (e**x - 1 - x)/(x**2)
```

Question 2.– Donner l'instruction qui en calcule la limite lorsque $x \to 0$.

```
limit(f, x=0)
```

Exercice 2. Debug.

On souhaite écrire une fonction `valuation(a, p)` qui prend en entrée deux entiers `a` et `p` strictement supérieurs à 1, et qui retourne le plus grand entier n tel que p^n divise a.

La fonction suivante a été écrite :

```
1  def valuation(p, a):
2      n=0
3      while (a % p == 0)
4          a //= p
5          n += 1
6      return n
```

En exécutant `valuation(2, 40)`, le programme retourne :

```
1      while (a % p == 0)
2                      ^
3  SyntaxError: invalid syntax
```

Question 1.– Expliquez quel est le problème, et comment corriger le programme pour qu'il renvoie la bonne valeur.

Le problème est une erreur de syntaxe, il manque un " :" à la fin de la ligne 3. Le code devient :

```
1  def valuation(p, a):
2      n=0
3      while (a % p == 0):
4          a //= p
5          n += 1
6      return n
```

Exercice 3. Un tri particulier.

Question 1.– Écrire une fonction `trouve_plus_proche(L, x)` qui prend en entrée une liste non-vide d'entiers `L` et un entier `x`, et qui retourne l'élément de `L` qui est le plus proche (en valeur absolue) de `x`.

Par exemple, si `L = [0, 7, 19, 4]` et `x = 10`, la fonction retournera l'entier `7`.

```
def trouve_plus_proche(L, x):
    resultat = L[0]
    ecart = abs(L[0] - x)
    for a in L:
        ecart_tmp = abs(a - x)
        if ecart_tmp < ecart:
            ecart = ecart_tmp
            resultat = a
    return a
```

Question 2.– Écrire une fonction `trie_selon_distance(L, x)` qui prend en entrée une liste non-vide d'entiers `L` et un entier `x`, et qui retourne la liste des éléments de `L`, triée de manière croissante selon leur distance à `x`.

Par exemple, si `L = [0, 7, 19, 4]` et `x = 10`, la fonction retournera la liste `[7, 4, 19, 0]`.

```
def trie_selon_distance(L, x):
    n = len(L)
    resultat = []
    for i in range(n):
         s = trouve_plus_proche(L, x)
         resultat.append(s)
         j = 0
         while L[j] != s:
             j += 1
         L.pop(j)
     return resultat
```

Remarque : on pouvait aussi modifier la fonction `trouve_plu_proche` pour qu'elle retourne également l'indice `j`.

Exercice 4. Résolution d'un système linéaire.

Question 1.– Donner la série d'instructions qui permettent de décrire l'ensemble des solutions rationnelles du système d'équations suivant :

$$\begin{cases} 7x - 3y + z = 0 \\ -x + 9y + 2z = 4 \end{cases}$$

Les inconnues sont (x, y, z).

```
M = matrix(QQ, [[7, -3, 1], [-1, 9, 2]])
b = vector(QQ, [0, 4])
solution_particuliere = M.solve_right(b)
espace_vectoriel = M.right_kernel()
```

L'ensemble des solutions est $\{\texttt{solution\_particuliere} + v, \quad v \in \texttt{espace\_vectoriel}\}$.

On peut accéder à une base de l'espace vectoriel `espace_vectoriel` en tapant `espace_vectoriel.basis()`

Exercice 5. Tangente à une courbe.

Question 1.– Écrire une fonction `tangente(f, a)` qui prend en entrée une expression symbolique `f` d'une fonction, et une valeur `a` dans le domaine de définition de la fonction, et qui retourne l'équation de la tangente à la courbe définie par la fonction `f` au point d'abscisse `a`.

La valeur de retour de la fonction sera une **expression symbolique** en la variable `x`. On supposera que `x` a déjà été déclarée. On supposera également de la fonction mathématique f est de classe $\mathcal{C}^1$ autour de a.

Par exemple, si $f(x) = x^2$ et $a = -1$, alors la fonction retournera $-2x - 1$.

```
def tangente(f, a):
    u = diff(f, x)(x=a)
    v = f(x=a)
    return u*(x-a) + v
```

Exercice 6. Calcul des termes d'une suite récurrente.

Soit u_n la suite récurrente définie par :

$$u_0 = 1, \quad u_1 = -1, \quad u_n = (u_{n-1})^2 + u_{n-2}$$

Question 1.– Écrire une fonction `calcule_u(n)` qui prend en entrée un entier naturel `n`, et qui retourne la valeur de u_n.

Par exemple, si $n = 2$, alors la fonction retournera $u_2 = (-1)^2 + 1 = 2$.

```
    def calcule_u(n):
        u0 = 1
        if (n == 0):
            return u0
        u1 = -1
        for i in range(1, n):
            tmp = u0
            u0 = u1
            u1 = u1**2 + tmp
        return u1
```

Exercice 7. Polynômes.

Question 1.– Donner la ou les instructions qui déclarent :

- l'anneau de polynômes sur les entiers relatifs, que l'on notera `R`
- l'indéterminée `X` de cet anneau de polynôme

```
    R = PolynomialRing(ZZ, "X")
    X = R.gen()
```

Question 2.– Écrire une fonction `polynome_aleatoire(degre, b)` qui prend en entrée deux entiers strictement positifs `degre` et `b`, et qui retourne un polynôme de degré au plus égal `degre`, dont tous les coefficients sont des entiers relatifs, tirés uniformément entre $-b$ et b (compris).

```
    def polynome_aleatoire(degre, b):
        P = 0
        for i in range(degre+1):
            coefficient = randrange(-b, b+1)
            P = P + coefficient * X**i
        return P
```

Exercice 8. Calcul sur des matrices.

Question 1.– Écrire une fonction `matrice_speciale(n)` qui prend en entrée un entier `n` strictement supérieur à 1, et qui retourne la matrice carrée de taille $(n \times n)$, définie sur les rationnels :

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ \vdots & & & & \ddots & 1 \\ 0 & \dots & \dots & \dots & \dots & 0 \end{pmatrix}$$

Par exemple, pour $n = 3$, la fonction retournera la matrice $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$.

```
    def matrice_speciale(n):
        L = []
        for i in range(n):
            ligne = [0 for j in range(n)]
            if i != n-1:
                ligne[i+1] = 1
            L.append(ligne)
        return matrix(L)

```

Une matrice carrée $\boldsymbol{M}$ de taille $n \times n$ est dite nilpotente d'indice $r \geq 1$ si $\boldsymbol{M}^r = \mathbf{0}$ et $\boldsymbol{M}^{r-1} \neq \mathbf{0}$. On peut démontrer que si une matrice est nilpotente, alors son indice r vérifie $r \leq n$.

Question 2.– Écrire une fonction `est_nilpotente(M)` qui prend en entrée une matrice `M`, et qui retourne l'indice r de nilpotence de `M` si `M` est nilpotente, et 0 si elle ne l'est pas.

```
    def est_nilpotente(M):
        r = 1
        while r <= n:
            if M**r == 0:
                return r
            r += 1
        return 0

```