Université Paris 8 — Année 2021–2022

Licence Mathématiques

Logiciel de calcul formel — Interrogation écrite — Sujet B

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. Un tri particulier.

Question 1.– Écrire une fonction `mot_plus_court(L)` qui prend en entrée une liste non-vide de chaînes de caractères `L`, et qui retourne le mot le plus court de `L`. Si plusieurs mots de la liste ont même longueur minimale, on pourra retourner n'importe lequel de ces mots.

Par exemple, si `L = ["logiciel", "de", "calcul", "formel"]`, alors la fonction retournera `"de"`.

```python
def mot_plus_court(L):
    resultat = L[0]
    longueur = len(L[0])
    for a in L:
        longueur_tmp = len(a)
        if longueur_tmp < longueur:
            longueur = longueur_tmp
            resultat = a
    return a
```

Question 2.– Écrire une fonction `trie_selon_longueur(L)` qui prend en entrée une liste non-vide de chaînes de caractères `L`, et qui retourne la liste des éléments de `L`, triée de manière croissante selon leur longueur.

Par exemple, si `L = ["logiciel", "de", "calcul", "formel"]`, alors la fonction pourra retourner la liste `["de", "calcul", "formel", "logiciel"]`.

```python
def trie_selon_longueur(L):
    n = len(L)
    resultat = []
    for i in range(n):
         mot = mot_plus_court(L)
         resultat.append(mot)
         j = 0
         while L[j] != mot:
             j += 1
         L.pop(j)
     return resultat
```

Exercice 2. 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 :

$$\begin{cases} -7a + 3b + 2c = 1 \\ 4a - 11b + c = 0 \end{cases}$$

Les inconnues sont (a, b, c).

```
M = matrix(QQ, [[-7, 3, 2], [4, -11, 1]])
v = vector(QQ, [1, 0])
solution_particuliere = M.solve_right(v)
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 3. Polynômes.

Question 1.– Donner la ou les instructions qui déclarent l'anneau de polynômes sur les entiers relatifs (que l'on notera `R`), et 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 4. Debug.

On souhaite écrire une fonction `est_arithmetique(L, r)` qui prend en entrée une liste `L` et un entier `r`, et qui retourne si `L` contient les premiers termes d'une suite arithmétique de raison `r`.

La fonction suivante a été écrite :

```
def est_arithmetique(L, r):
    n = len(L)
    for i in range(n):
        if (L[i+1] - L[i] != r):
            return False
    return True
```

En exécutant `est_arithmetique([2, 5, 8, 11], 3)`, le programme retourne :

```
    if (L[i+1] - L[i] != r):
IndexError: list index out of range
```

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

Le problème est que l'on souhaite accéder à un élément de la liste qui n'existe pas. En effet, pour $i = n - 1$, on souhaite accéder à `L[i+1]`, c'est-à-dire `L[n]`, qui n'existe pas. IL faut donc s'arrêter une étape avant. Le code devient :

```
def est_arithmetique(L, r):
    n = len(L)
    for i in range(n-1):
        if (L[i+1] - L[i] != r):
            return False
    return True
```

Exercice 5. Fonctions et sommes.

Question 1.– Déclarer une expression symbolique `u` en la variable `n` qui vaut $u(n) = \dfrac{1}{n2^n}$.

```
var('n')
u = 1/(n*2**n)
```

Question 2.– Donner l'instruction qui calcule $\sum\limits_{n=1}^{+\infty} u(n)$.

```
sum(u, n, 1, oo)
```

Exercice 6. 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 comme :

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

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

```
    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
            else:
                ligne[0] = 1
            L.append(ligne)
        return matrix(L)

```

Une matrice carrée $\boldsymbol{M}$ de taille $n \times n$ est dite **circulante** si toute ligne de la matrice se déduit de la précédente par une rotation vers la droite des coefficients. Autrement dit, une matrice circulante a la forme suivante :

$$A = \begin{pmatrix} x_1 & x_2 & x_3 & \dots & \dots & x_n \\ x_n & x_1 & x_2 & x_3 & \dots & x_{n-1} \\ x_{n-1} & x_n & x_1 & x_2 & \dots & x_{n-2} \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ x_3 & & & \ddots & \ddots & x_2 \\ x_2 & x_3 & \dots & \dots & x_n & x_1 \end{pmatrix}$$

Question 2.– Écrire une fonction `est_circulante(M)` qui teste si une matrice `M` est circulante. La fonction retournera le booléen associé.

```
    def est_circulante(M):
        n = M.nrows()
        return all(M[i][j] == M[i+1 % n][j+1 % n]
                   for i in range(n) for j in range(n))

```

Exercice 7. Calcul des termes d'une suite récurrente.

Soit u_n la suite récurrente définie par :

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

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 $u_2 = -2 \times (-1) + 1^2 = 3$.

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

Exercice 8. Développement limité d'ordre 1 en un point.

Question 1.– Écrire une fonction `developpement_ordre_1(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 le développement limité d'ordre 1 de `f` autour du point d'abscisse `a`.

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

Par exemple, si $f(x) = e^x$ et $a = 0$, alors la fonction retournera $1 + x$.

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