Devoir maison n°8

A rendre le 04/03.
Vous pouvez rendre une copie pour deux si et seulement si chacun participe à la recherche ET à la rédaction.

Préambule

Ce devoir mêle des questions mathématiques et des questions informatiques. Vous disposez d'un fichier **dm8.py** à compléter.

Le module `fractions`

Les objets `Fraction` permettent d'effectuer des calculs exacts sur les rationnels. La création de `Fraction` est très simple

```
from fractions import Fraction
f = Fraction(2, 3)  # f est 2/3
f2 = Fraction(3, 4)
f + f2, f*f2, 6*f  # calcule la somme, le produit, le produit par un entier
f3 = f2/7 # f est f2/7 ie 3/28
# cette dernière commande est la même que f3 = Fraction(f2, 7)
```

Exercice 1
Dans cet exercice, pour $P, Q \in \mathbb{R}[X]$, on définit :

$$< P, Q >= \int_{-1}^{1} P(t)Q(t)\mathrm{d}t$$

1. Pour un polynôme $P = \sum_{k=0}^{n} a_i X^i$, calculer en fonction des coefficients $a_0, \dots, a_n$ l'intégrale $\int_0^1 P(t)\mathrm{d}t$.
2. Vérifier que $< ., . >$ est un produit scalaire sur $\mathbb{R}[X]$. On en déduit que c'est également un produit scalaire sur $\mathbb{R}_n[X]$ pour tout $n \in \mathbb{N}$.
   On notera que la norme d'un polynôme P est définie par $\|P\|^2 =< P, P >= \int_0^1 P^2(t)\mathrm{d}t$.
3. Compléter la fonction `scalaire(P, Q)` qui calcule le produit scalaire de deux polynômes donnés sous formes de listes. Avec les notations de la question 1, le polynôme P est encodé en python par la liste $[a_0, \dots, a_n]$ de longueur $n+1$.
   On pensera à utiliser des objets `Fraction` pour chaque terme de la somme, et ainsi on retournera un objet `Fraction` quand P, Q sont à coefficients entiers (ou donnés sous forme de fractions).
4. Appliquer, à la main, l'algorithme de Gram-Schmidt à $\mathcal{B}_2 = (1, X, X^2) = (P_0, P_1, P_2)$ pour trouver une base orthogonale et échelonnée en degrés de $\mathbb{R}_2[X]$. On notera (Q_0, Q_1, Q_2) la base trouvée.
5. On souhaite appliquer l'algorithme de Gram-Schmidt précédent à la base $B_n = (1, X, \dots, X^n) = (P_i)_{i\in[\![0,n]\!]}$ (où $n \in \mathbb{N}$, $n \geqslant 3$) pour obtenir $(Q_0, \dots, Q_n)$. On cherche d'abord Q_3 sous la forme $Q_3 = P_3 - a_{3,0}P_0 - a_{3,1}P_1 - a_{3,2}P_2$ où $a_{3,j} \in \mathbb{R}$ pour $j \in [\![0, 2]\!]$ est à trouver. Exprimer $a_{3,j}$ à l'aide de produits scalaires pour $j \in [\![0, 2]\!]$.
   Donner les coefficients de Q_3. On pourra utiliser python et la fonction `scalaire` en particulier.
6. Dans le cas général et pour $k \in [\![1, n]\!]$ fixé, on cherche Q_k sous la forme $Q_k = P_k - \sum_{j=0}^{k-1} a_{k,j}Q_j$ où les $a_{k,j} \in \mathbb{R}$ sont à trouver pour $j \in [\![0, k-1]\!]$. Pour $j \in [\![0, k-1]\!]$, exprimer $a_{k,j}$.
7. En python, une famille $F = (R_0, \dots, R_n)$ de $n+1$ polynômes est encodé par la liste des polynômes, ie par une liste de listes. Par exemple la famille B_2 est encodée par

   ```
   B2 = [[Fraction(1, 1)], [0, Fraction(1, 1)], [0, 0, Fraction(1, 1)]]
   ```

   (a) Compléter la fonction `base_canonique`. Vérifier que la valeur de retour pour l'argument 2 est bien la base de l'exemple (ou, éventuellement, chaque liste ne contient que des objets `Fraction`).

   (b) On souhaite implémenter l'algorithme de Gram-Schmidt pour une famille F.
   Le principe est de construire une famille $Q = (Q_0, \dots, Q_n)$ sous la forme d'une liste. On notera la variable locale `Q` également. En python, une fois construit $Q_0, \dots Q_k$ (que l'on stockera au fur et à mesure dans `Q`), le polynôme Q_i sera accessible par `Q[i]`.
   Compléter la fonction `gram_schmidt`. On pourra la tester avec

`gram_schimdt(base_canonique(2))`

Pour obtenir la base de la question 4.

Remarque : il n'est pas difficile d'adapter notre algorithme pour s'adapter à d'autres produits scalaires. Il suffit de changer la fonction correspondante, ou, encore mieux, de passer la fonction calculant le produit scalaire comme paramètre à `gram_schmidt`

Exercice 2

Dans cet exercice, se place dans $\mathbb{R}^n$ munit du produit scalaire usuel.

On considère n points de $\mathbb{R}^2$, $M_i : \begin{pmatrix} x_i \\ y_i \end{pmatrix}$ pour $i \in [\![1, n]\!]$, et on cherche une droite $\mathcal{D} : y = ax + b$ qui serait une "meilleure approximation affine" du nuage de point $\{M_i,\ i \in [\![1, n]\!]\}$. Pour cela on cherche à minimiser la quantité (dépendant de a et b).

$$S(a, b) = \sum_{i=1}^{n} (y_i - (ax_i + b))^2$$

qui représente la somme des carrés des distances obtenues par projection verticale. C'est la méthode des moindres carrés.

1. Première méthode. On considère $X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$, $Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$ et $U = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^n$.
   (a) Pour $a, b \in \mathbb{R}$ fixés, exprimer $S(a, b)$ sous forme d'une norme faisant intervenir X, Y, U.
   (b) Montrer que $S(a, b)$ est minimal lorsque $aX + bU$ est le projeté orthogonal de Y sur $F = \text{Vect}(U, V)$.
   (c) En déduire que le couple (a, b) cherché est solution du système

$$\begin{cases} a\sum_{i=1}^{n} x_i + nb = \sum_{i=1}^{n} y_i \\ a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \end{cases}$$

   (d) Montrer que si les M_i n'ont pas tous la même abscisse, alors le système de la question précédente est un système de Cramer, et donner son unique solution. [1]
   (e) Compléter la fonction `regression`. On pourra utiliser `numpy.linalg.solve` pour résoudre le système considéré.
2. Deuxième méthode. On considère la fonction $S : \begin{cases} \mathbb{R}^2 & \to & \mathbb{R} \\ (a, b) & \mapsto & \sum_{i=1}^{n} (y_i - (ax_i + b))^2 \end{cases}$.
   (a) Montrer que S est de classe $\mathcal{C}^1$ sur l'ouvert $\mathbb{R}^2$.
   (b) Calculer $\frac{\partial S}{\partial a}$ et $\frac{\partial S}{\partial b}$.
   (c) On suppose encore que les M_i n'ont pas tous la même abscisse. Montrer que S possède une unique point critique en un point à déterminer. On admet provisoirement que ce point critique correspond à un minimum.
3. Troisième méthode (question bonus). Cette fois on considère que les M_i sont d'abscisses 2 à 2 distinctes.
   (a) Montrer qu'il existe un unique polynôme $L \in \mathbb{R}_{n-1}[X]$ tel que $\forall i \in [\![1, n]\!] L(x_i) = y_i$. Comment appelle-t-on ce polynôme ?
   (b) On munit $\mathbb{R}_{n-1}[X]$ du produit scalaire $< P, Q >= \sum_{i=1}^{n} P(x_i)Q(x_i)$. On admet qu'il s'agit bien d'un produit scalaire. [2]
   Ecrire $S(a, b)$ sous forme d'une norme au carré (pour ce produit scalaire).
   (c) On en déduit que l'on peut interpréter $aX + b$ comme le projeté orthogonal de L sur $\mathbb{R}_1[X]$. Comment faire pour obtenir plutôt la parabole d'équation $y = ax^2 + bx + c$ qui soit la plus proche possible du nuage de points ?

1. On pourra aller réviser l'inégalité de Cauchy-Schwartz, ainsi que son cas d'égalité.
2. Voir le TD pour un exemple de démonstration.