2TSI. TD PYTHON 10

Tracé de surface et algorithme de Schmidt

1. Avec Python, tracer $f : (x, y) \mapsto \sum_{k=0}^{4}(k^4 - xk^3 - y)^2$ sur $[0, 10] \times [-20, 0]$.
   Observer l'existence d'un minimum de F et affiner l'intervalle.
   On pourra trouver en bonus la valeur exacte de ce minimum avec *sympy*.

2. Montrer que $\langle P, Q\rangle = \sum_{k=0}^{4} P(k)Q(k)$ est un produit scalaire sur $\mathbb{R}_4[X]$.

3. En déduire l'existence de $A = \inf_{(x,y)\in\mathbb{R}^2} \sum_{k=0}^{4}(k^4 - xk^3 - y)^2$ et le calculer (avec Python).
   Indication : il faut faire le lien entre le produit scalaire de la question **2)** *et la fonction de la question* **1)**. *L'existence de A est due à un résultat de cours qu'il faut énoncer correctement. Pour le calcul de A, on utilise bien entendu le logiciel. On définit la fonction produit scalaire $\langle\ ,\ \rangle$ par phi (pour l'écrire simplement sous Python) puis on rentre les polynômes 1 et X^3 en les baptisant $u1$ et $u2$. Puis, on construit une base orthonormale $(w1, w2)$ de $F = \mathrm{Vect}\,(1, X^3)$ à partir de Gram-Schmidt. Puis on calcule ZZ, la projection orthogonale de $Z = X^4$ sur F. Puis on détermine $\|Z - ZZ\|^2$. C'est A cherché.*

Quelques rappels utiles pour l'exercice.

• Il faut se souvenir de la façon dont on trace une surface avec Python.
Pour ***tracer une surface***, d'équation $z = f(x, y)$, on réalise d'abord une grille en (x, y) en supposant $x \in [a, b]$ et $y \in [c, d]$ avec un pas h puis on calcule les valeurs de z correspondants aux points de cette grille. On fait ensuite le tracé avec la fonction *plot_surface* issue du module *mpl_toolkits.mplot3d*. On tape :

>>> import numpy as np ; import matplotlib.pyplot as plt
>>> from mpl_toolkits.mplot3d import Axes3D
>>> X = np.arange(a, b, h) ; Y = np.arange(c, d, h)
>>> X, Y = np.meshgrid(X, Y) ; Z = f(X, Y)
>>> ax = Axes3D(plt.figure( )) ; ax.plot_surface(X, Y, Z) ; plt.show( )

• Il faut se souvenir de la façon dont on trace une ligne de niveau avec Python.
Pour ***tracer des lignes de niveau*** $\{(x, y) \in [a, b] \times [c, d],\ f(x, y) = k\}$, où k est un réel (qui peut prendre plusieurs valeurs), on fait une grille en X et en Y de pas h sur laquelle on calcule les valeurs de f. On emploie ensuite *contour* du module *matplotlib.pyplot* en mettant dans une liste L les valeurs de k pour lesquelles on désire le tracé. On tape :

>>> import numpy as np ; import matplotlib.pyplot as plt
>>> X = np.arange(a, b, h) ; Y = np.arange(c, d, h)
>>> X, Y = np.meshgrid(X, Y)
>>> Z = f(X, Y) ; plt.axis('equal') ; plt.contour(X, Y, Z, L) ; plt.show( )

• Il faut se souvenir de la façon dont on fait des opérations sur les polynômes avec le logiciel Python.
On peut rentrer un polynôme comme une fonction. On peut aussi utiliser la fonction *Polynomial* du sous-module *numpy.polynomial* et un certain nombre d'attributs associés. On commence donc par taper :

>>> from numpy.polynomial import Polynomial

1) Pour ***créer un polynôme***, on liste ses coefficients par ordre de degré croissant.
Ainsi pour $p = a0 + a1X + a2X^2 + a3X^3$, on tape : $p = Polynomial([a0, a1, a2, a3])$.
L'attribut *coef* donne accès aux coefficients ordonnés par degré croissant.
Ainsi, si l'on tape : *p.coef*, on obtient $array([a0., a1., a2., a3.])$