⋆ Fonctions de plusieurs variables ⋆

Compétences attendues

- ✓ Tracer le graphe d'une fonction de plusieurs variables ;
- ✓ Tracer un champ de gradients et des lignes de niveau d'une fonction de plusieurs variables.

Liste des commandes Python exigibles aux concours

- Dans la librairie `numpy` : `linspace`, `meshgrid` ;
- Dans la librairie `matplotlib.pyplot` : `show`, `contour`, `quiver` ;
- Avec la fonction `Axes3D` : `ax.plot_surface`.

1 Représentation graphique

1.1 Saisie de f

Dans toute la suite, f désigne une fonction de $\mathbb{R}^2$ dans $\mathbb{R}$. Pour définir la fonction f sur Python, on utilise le code suivant :

```
def f(x,y) :
    return ...
```

1.2 Graphe de f

Afin de représenter une fonction de deux variables à l'aide de Python, nous aurons besoin d'importer les librairies suivantes :

```
import numpy as np
import matplotlib.pyplot as plt
```

Nous aurons également besoin de la fonction `Axes3D` de la librairie `mpl_toolkits.mplot3d`, qu'on importe comme suit :

```
from mpl_toolkits.mplot3d import Axes3D
ax = Axes3D(plt.figure())
```

Remarque 1.1. Selon la version de Python utlisée, ces deux dernières lignes de commandes peuvent être remplacées par la suivante :

```
ax = plt.axes(projection = '3d')
```

Définition 1.1

Soient `x` et `y` des vecteurs de taille respective n et m. L'instruction

`X,Y=np.meshgrid(x,y)`

permet de construire le maillage $((x_i, y_j))_{(i,j)\in[\![1,n]\!]\times[\![1,m]\!]}$.

Méthode 1.1 [Comment tracer le graphe d'une fonction f ?]

Pour tracer la représentation graphique de f sur $[a, b] \times [c, d]$, on procèdera comme suit :

1. On crée deux vecteurs `x` et `y` découpant les intervalles $[a, b]$ et $[c, d]$ en n petits intervalles de même longueur comme suit :
   ```
   x = np.linspace(a,b,n)
   y = np.linspace(c,d,n)
   ```
2. On crée ensuite un maillage $((x_i, y_j))_{1\leq i,j\leq n}$ du domaine $[a, b] \times [c, d]$ avec la commande :
   `X,Y=np.meshgrid(x, y)`
3. On commande enfin le tracé avec l'instruction :
   ```
   ax.plot_surface(X, Y, f(X, Y))
   plt.show ()
   ```

Exercice 1 Représenter le graphe de la fonction $f : (x, y) \mapsto \sin(x)\sin(y)$ sur $[-6, 6] \times [-6, 6]$. On pourra tester successivement $n = 5, 20, 50, 200$ pour définir les vecteurs `x` et `y`.

Remarque 1.2.

- La représentation graphique d'une fonction de deux variables n'est autre qu'une succession de rectangles définis par le

maillage de la grille. On doit donc veiller à ce que ce maillage soit suffisamment fin pour donner l'illusion d'une surface lisse. Dans la suite, on prendra $n = 50$ qui est un bon compromis entre la qualité de la représentation du graphe de f et la complexité de calcul pour Python.

- On peut faire tourner le graphe afin de l'orienter à sa guise à l'aide de la souris.
- On peut changer de couleurs à l'aide de l'argument cmap : `ax.plot_surface(X, Y, f(X, Y), cmap='jet')` Si les couleurs ne sont toujours pas à vôtre goût, vous pouvez remplacer `'jet'` par `'cool'`, `'winter'`, `'spring'`, `'summer'`, `'autumn'`,`'hot'`, `'terrain'`, `'prism'`,...

1.3 Lignes de niveau

Définition 1.2

Soit `X, Y` un maillage du domaine $[a, b] \times [c, d]$ et `f` une fonction prenant deux arguments réels.

Les commandes

`plt.contour(X,Y,f(X,Y),N)` ou `plt.contour(X,Y,f(X,Y),T)`

tracent les lignes de niveau de la fonction f, avec au choix comme dernier argument de la fonction `plt.contour` :

- un entier `N` : dans ce cas, on obtient `N-1` lignes de niveau entre les valeurs minimales et maximales de f sur le maillage.
- un tableau `T` : dans ce cas, on obtient les lignes de niveau associées aux valeurs contenues dans le tableau `T`.

Remarque 1.3. Pour que les lignes de niveau soient représentées en 2D, il est nécessaire de mettre en commentaire (ou de ne pas exécuter) la commande :

```
ax=Axes3D(plt.figure())
```

Exercice 2 On considère la fonction $f : (x, y) \in \mathbb{R}^2 \mapsto \left(x^3 + y^2\right) e^{-\left(x^2+y^2\right)}$. Tracer le graphe de f ainsi que n lignes de niveau pour $n = 5$ puis 10 , sur le domaine $[-3, 3] \times [-3, 3]$.

2 Calcul différentiel du premier ordre

2.1 Dérivées partielles, gradient

Exercice 3 On considère la fonction $f : (x, y) \in \mathbb{R}^2 \mapsto \left(x^3 + y^2\right) e^{-\left(x^2+y^2\right)}$.

1. Justifier que f est de classe $\mathscr{C}^1$ sur $\mathbb{R}^2$, puis déterminer ses dérivées partielles $\partial_1 f$ et $\partial_2 f$.
2. Définir en Python les fonctions d1f et d2f qui donnent les dérivées partielles $\partial_1 f$ et $\partial_2 f$ respectivement.

Étant donné que le gradient $\nabla f(x, y)$ est un vecteur dirigé dans le sens de la plus forte pente (en pointant vers la montée), il peut être utile de visualiser le champ de gradients associé à une fonction $f : \mathbb{R}^2 \to \mathbb{R}$, c'est-à-dire :

- on fixe un maillage de points $((x_i, y_j))_{(i,j)\llbracket 1,n\rrbracket\times\llbracket 1,m\rrbracket}$ du plan ;
- en chaque point (x_i, y_j) du maillage, on représente le vecteur gradient $\nabla f\,(x_i, y_j)$.

On utilise pour cela la commande `quiver`.

Définition 2.1

Soient `X, Y` et `VX, VY` deux maillages de points de même taille $n \times m$.

La commande `plt.quiver(X,Y,VX,VY)` trace en chaque point `(X[i],Y[j])` du plan le vecteur de coordonnées `(VX[i],VY[j])` pour tout $(i, j) \in \llbracket 1, n\rrbracket \times \llbracket 1, m\rrbracket$.

Remarque 2.1. Les vecteurs représentés n'ont pas pour composantes `(VX[i],VY[j])`. Python applique un coefficient de proportionnalité qui modifie les normes (mais pas les orientations).

Méthode 2.1 [Comment représenter un champ de gradients d'une fonction]

Pour tracer le champ de gradients associé à une fonction $f : [a, b] \times [c, d] \to \mathbb{R}$, on procèdera ainsi :

1. Si ce n'est pas le cas, mettre en commentaire la commande :
   `ax=Axes3D(plt.figure())`
2. On crée le maillage `X, Y` du domaine $[a, b] \times [c, d]$ (en prenant garde de ne pas le choisir trop serré) à l'aide des commandes :

```
x = np.linspace(a,b,n)
y = np.linspace(c,d,n)
X,Y = np.meshgrid(x,y)
```

3. On déclare les fonctions `d1f` et `d2f` qui donnent les dérivées partielles $\partial_1 f$ et $\partial_2 f$.
4. On obtient alors le champ de vecteurs de f avec l'instruction :

```
plt.quiver(X,Y,d1f(X,Y),d2f(X,Y))
plt.show()
```

Exercice 4 Tracer sur un même graphique le champ de gradients et les lignes de niveau de $f : (x,y) \in \mathbb{R}^2 \mapsto \left(x^3+y^2\right) e^{-\left(x^2+y^2\right)}$. On utilisera pour cela le réseau défini par $x_i = -3+i$ et $y_j = -3+j$ pour $0 \leq i,j \leq 6$.

Interpréter le résultat : direction et norme du gradient, repérer le(s) point(s) critique(s) éventuel(s) et leur nature, positions relatives du gradient et des lignes de niveau. En particulier, quel résultat évoqué en cours retrouvez vous ?

2.2 Plan tangent

Méthode 2.2 [Comment tracer le plan tangent au graphe d'une fonction en un point ?]

Pour représenter le plan affine tangent au graphe de f en (x_0,y_0), on définit sur Python la fonction affine :

$$t_{(x_0,y_0)} : (x,y) \mapsto f(x_0,y_0) + \partial_1(f)(x_0,y_0) \times (x-x_0) + \partial_2(f)(x_0,y_0) \times (y-y_0)$$

On trace sa représentation graphique sur le même graphe que f (au moins au voisinage de $\xi(x_0,y_0)$)

Exercice 5 On considère la fonction $f : (x,y) \mapsto \left(x^3+y^2\right) e^{-\left(x^2+y^2\right)}$ sur $[-3,3]\times[-3,3]$. Tracer la représentation graphique de f ainsi que celle des plans affines tangents aux points $(0,0)$, $(0,1)$ puis $(1,2)$.