TPB5 - Graphes de fonctions de plusieurs variables

Dans tout le TP, on importe les modules suivants :

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

1 Généralités

Dans ce TP, nous allons revenir sur les fonctions de plusieurs variables et nous allons utiliser `numpy` et `matplotlib` pour représenter ces fonctions. Pour des raisons évidentes, nous nous limiterons à des fonctions définies sur $\mathbb{R}^2$ (ou une partie de $\mathbb{R}^2$) et à valeurs dans $\mathbb{R}$ représentées donc par un graphe tridimensionnel.

Exercice 1 - Graphe d'une fonction de $\mathbb{R}^2$ ⋆⋆

Tracer une fonction de $\mathbb{R}^2$ en `Python` est relativement technique. Vous avez sans doute remarquer l'importation nouvelle :

```
from mpl_toolkits.mplot3d import Axes3D
```

C'est ce module qui va permettre de faire le tracé 3D. Il s'utilise de la manière suivante :

```
ax = Axes3D(plt.figure())
ax.plot_surface(X,Y,Z)
```

où `X`, `Y` et `Z` sont trois tableaux `numpy` contenant les coordonnées 3D des points à utiliser pour dessiner la surface. Cela pose quelques difficultés techniques que nous allons résoudre une à une :

1. On va chercher à représenter la fonction : $f : (x, y) \mapsto \sin(x)\sin(y)$. Comme pour le tracé habituel d'une fonction à une variable, on va générer des valeurs d'antécédents que l'on aimerait utiliser :

   ```
   x = np.linspace(-5,5,11)
   y = np.linspace(-5,5,11)
   ```

   À quoi ressemble la grille que l'on souhaite utiliser d'après le code précédent ? La dessiner dans le plan. Combien de points possède-t-elle ? Combien de nombres contiennent les tableaux `x` et `y` ? Commenter.
2. On utilise la commande :

   ```
   x_mesh,y_mesh = np.meshgrid(x,y)
   ```

   Observer le contenu de `x_mesh` et `y_mesh` et expliquer comment cela résout le problème précédent.
3. On utilise enfin le code :

   ```
   z_mesh = np.sin(x_mesh)*np.sin(y_mesh)
   ```

   Utiliser `x_mesh`, `y_mesh` et `z_mesh` avec le module `Axes3D` pour tracer la fonction.
4. Quel(s) paramètre(s) faut-il ajuster pour avoir un tracé plus fin ?

On peut aussi les commandes :

```
my_cmap = plt.get_cmap('plasma')
ax.plot_surface(x_mesh,y_mesh,z_mesh,cmap=my_cmap)
```

pour avoir plus de couleurs.

2 Lignes de niveau et gradients

Exercice 2 - Lignes de niveau ⋆

Comparativement au tracé du graphe, le tracé des lignes de niveau est plus simple même si des problématiques similaires émergent. En effet, il faut utiliser la commande `plt.contour(X,Y,Z,vals)` où `X`, `Y` et `Z` sont encore trois tableaux `numpy` contenant les coordonnées 3D des points de la surface de la fonction et `vals` est un tableau `numpy` contenant les valeurs pour lesquelles on veut tracer les courbes de niveau.

1. On utilise :

```
vals = np.linspace(-1,1,21)
```

   Quelles vont être les lignes de niveau représentées ?
2. Reprendre la fonction de l'exercice précédent et tracer ses lignes de niveau.

Exercice 3 - Gradient ⋆⋆

Nous allons superposer au tracé des lignes de niveau celui du gradient de la fonction en différents points. La fonction à utiliser est `plt.quiver(X,Y,U,V)` où `X`, `Y` sont deux tableaux contenant les coordonnées des points pour lesquels on trace le gradient et `U`, `V` sont deux tableaux contenant les deux composantes du gradient en chacun de ces points.

1. Calculer le gradient en tout point de $f : (x, y) \mapsto \sin(x)\sin(y)$.
2. Expliquer, recopier puis exécuter le code suivant :

```
x = np.linspace(-5,5,110)
y = np.linspace(-5,5,110)
x_mesh,y_mesh = np.meshgrid(x,y)

my_cmap = plt.get_cmap('plasma')
vals = np.linspace(-1,1,21)
plt.contour(x_mesh,y_mesh,z_mesh,vals,cmap=my_cmap)

u_mesh = np.cos(x_mesh)*np.sin(y_mesh)
v_mesh = np.sin(x_mesh)*np.cos(y_mesh)
plt.quiver(x_mesh,y_mesh,u_mesh,v_mesh)

plt.show()
```

3. Commenter le graphique obtenu.

3 Plan tangent

Exercice 4 - Un premier tracé ⋆

On va étudier la fonction $g : (x, y) \mapsto x^2 + y^2$. On utilise le code suivant pour la tracer :

```
x = np.linspace(-5,5,110)
y = np.linspace(-5,5,110)
x_mesh,y_mesh = np.meshgrid(x,y)

z_mesh = x_mesh*x_mesh + y_mesh*y_mesh

ax = Axes3D(plt.figure())
my_cmap = plt.get_cmap('plasma')
ax.plot_surface(x_mesh,y_mesh,z_mesh,cmap=my_cmap)
plt.show()
```

1. Calculer le gradient de g en tout point. Quelle est l'interprétation géométrique de ce gradient vis-à-vis des plans tangents à la courbe de g ?
2. Montrer que l'équation du plan tangent à la courbe de g au point $(3, 1)$ est $z = 6x + 2y - 10$.
3. On complète le code précédent en rajoutant avant `plt.show()` les lignes suivantes :

```
z_mesh2 = 6*x_mesh + 2*y_mesh - 10
ax.plot_surface(x_mesh,y_mesh,z_mesh2,cmap=my_cmap)
```

   Que font ces lignes ?