TP 1
Représentation de fonctions réelles de 1 et 2 variables

1 Les paquets python de la séance

Les paquets utiles pour représenter des fonctions sont
- `matplotlib` qui permet de faire tout type de visualisation
- `math` qui fournit les constantes et fonctions usuelles en maths
- `numpy` qui permet de représenter les discrétisations sous forme de vecteurs (ou de matrices) et d'effectuer des opérations sur ces variables.

Il est préférable en python d'importer uniquement ce qui est utile, et de renommer le paquets avec leurs surnoms usuels dans la communauté. On évitera les imports massifs avec `*` qui de plus ne permettent pas de distinguer des méthodes portant le même nom dans plusieurs paquets. Par exemple, il y a une fonction sinus `sin` à la fois dans `math` et dans `numpy`.

Pour cette séance, on fait les imports suivants.

```
import numpy as np
from matplotlib import pyplot as plt
from math import pi
```

2 Représentation de la fonction sinus cardinal

Rappelons que la fonction sinus cardinal est définie par $\mathrm{sinc}(x) = \dfrac{\sin(x)}{x}$ pour $x \neq 0$ et $\mathrm{sinc}(0) = 1$. Nous allons voir différentes façons de définir la discrétisation de x et différents rendus du tracé.

2.1 Discrétisation de l'espace

Dans tous les cas, nous allons définir une valeur minimum $x_{\min}$ et une valeur maximum $x_{\max}$ de l'intervalle de définition (ou plutôt de tracé) de la fonction. Deux cas de figure se présentent :
- soit on se donne un nombre de points de discrétisation et on utilise `np.linspace`,
- soit on se donne un pas de discrétisation et on utilise `np.arange`.

Par défaut le dernier point retourné par `np.linspace` est $x_{\max}$. Ceci peut être modifié avec l'option `endpoint = False`. On peut retourner le pas de discrétisation avec l'option `retstep = true`. Par défaut également le type des éléments retournés est `float`.

En ce qui concerne `np.arange`, sauf si on a bien calculé son coup, il n'y a pas de raison que $x_{\max}$ soit le dernier point. Le type des données est induit du type des arguments de la fonction.

➽ Discrétiser l'intervalle $I = [-10, 10]$ avec 101 points et avec un pas de 0.2. Observer les différences.

2.2 Définition des valeurs de la fonction

Les fonctions de `numpy` peuvent s'appliquer à tous les éléments d'un tableau. Attention : il existe une fonction `sinc` dans `numpy` mais elle est définie différemment. Celle-ci a néanmoins l'avantage de bien se comporter si $x = 0$.

➽ Définir l'image des discrétisations précédentes par la fonction sinus cardinal de deux manières : a) en reconstituant une fonction avec le sinus et une division, b) en utilisant la fonction prédéfinie de `numpy`. Constater à nouveau les différences.

2.3 Tracé des fonctions

La fonction la plus simple pour tracer une fonction est `plt.plot`. On peut lui adjoindre un large jeu d'arguments pour paramétrer le tracé. Les méthodes `plt.xlabel`, `plt.ylabel` et `plt.title` permettent de préciser la signification des axes des abcisses et des ordonnées et de donner un titre au graphe. Le paramètre `label` est une manière élégante d'associer à chaque courbe sa signification dont l'impression se fait grâce à `plt.legend`.

➽ Tracer une des courbes précédentes avec un trait continu bleu. Utiliser une discrétisation avec 21 points pour tracer sur le même graphe des points de calculs individuels rouges. Assortir le graphe d'un titre, de noms d'axes, et d'une légende.

3 Représentation d'une surface

On considère la fonction

$$f(x, y) = x^3 + 3xy^2 - 15x - 12y. \tag{1}$$

Pour la discrétisation d'une surface, il faut définir les points de discrétisation dans les deux directions, typiquement avec `np.linspace`, puis ensuite définir grâce à `np.meshgrid` deux matrices donnant respectivement les abscisses et les ordonnées de tous les points.

On décide de tracer deux figures côte-à-côte, ce qui se fait avec `plt.subplots`. On affiche la surface sur les axes courants (obtenus avec `plt.gca`) avec la méthode `plot_surface`. L'échelle des couleurs est associée à cette surface en utilisant `plt.colorbar`. Le choix des couleurs peut par exemple se définir avec `plt.get_cmap`.

Pour les lignes de niveau, on utilise la fonction `plt.contour` à laquelle on peut préciser les valeurs des contours à dessiner. On peut utiliser `plt.clabel` pour identifier les valeurs associées à chaque niveau. On peut comme pour les surfaces choisir la gamme de couleurs utilisées.

➽ Tracer la surface et les lignes de niveau de la fonction f sur deux graphes côte-à-côte. On utilisera le domaine de représentation $[-3, 3] \times [-3, 3]$ et une discrétisation de 40 points dans chaque direction. On assortira d'une échelle de couleurs. Soigner sa note artistique en utilisant une carte de couleurs flatteuse. Assortir bien sûr le graphe d'un titre, de noms d'axes.