Formulaire Modélisation Mouvements collectifs

Alexis Raulin–Foissac
alexis@raulin-foissac@univ-lyon1.fr

Contents

1 Python

Python est un langage de programmation populaire en science et en ingénierie. Deux bibliothèques essentielles sont :

- **NumPy** : permet de manipuler efficacement des tableaux de données.
- **Matplotlib** : permet de créer des graphiques et de visualiser des données.

On les importe généralement au début du code avec:

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

1.1 NumPy : Manipulation de tableaux

NumPy est une bibliothèque qui permet de manipuler des tableaux multidimensionnels avec une grande efficacité. Elle est largement utilisée pour les calculs scientifiques et l'analyse de données.

1.1.1 Fonctions utiles de NumPy

- **np.array(data)** : crée un tableau NumPy à partir d'une liste ou d'un autre objet itérable.
- **np.zeros((shape))** : crée un tableau rempli de zéros de la dimension spécifiée.
- **np.ones(shape)** : crée un tableau rempli de uns.
- **np.linspace(start, stop, num)** : génère un tableau de valeurs espacées linéairement.
- **np.random.uniform(valeur min, valeur max, shape)** : génère un tableau de valeurs aléatoires entre valeur min, valeur max de taille shape.
- **np.mean(array)** : calcule la moyenne des éléments d'un tableau.
- **np.std(array)** : calcule l'écart-type.
- **np.dot(array1, array2)** : effectue un produit scalaire entre deux tableaux.
- **np.sqrt(array), np.round(array), np.exp, np.cos, np.angle(nbr complexe), np.mean**: la plupart des fonctions mathématiques sont implémentés en numpy et les appeler sur un tableau revient à appliquer la fonction mathématique à chacun des éléments du tableau.

1.1.2 Slicing en NumPy

Le slicing en NumPy permet d'extraire des sous-tableaux à partir d'un tableau existant en utilisant la notation `array[start:stop:step]`.

- **A[start:stop]** : sélectionne les éléments entre les indices `start` et `stop-1`.
- **A[:n]** : sélectionne les `n` premiers éléments.
- **A[n:]** : sélectionne les éléments à partir de l'indice `n`.
- **A[::-1]** : inverse l'ordre des éléments.
- **A[start:stop:step]** : sélectionne des éléments avec un pas spécifique.

Exemple :

```
import numpy as np
A = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
print(A[2:7])     # [2 3 4 5 6]
print(A[:5])      # [0 1 2 3 4]
print(A[::2])     # [0 2 4 6 8]
print(A[::-1])    # [9 8 7 6 5 4 3 2 1 0]
```

Si le tableau possède plusieurs dimensions, une matrice (N*N) par exemple, il est possible d'effectuer du slicing sur les lignes et les colonnes en même temps:

- **A[:, 0]** ou **A[:, n-1]**: sélectionne toutes les lignes et uniquement la première colonne ou bien la n-eme colonne dans le second cas.
- **A[0, :]** : sélectionne toutes les colonnes et uniquement la première ligne.
- **A[2:, ::2]** : sélectionne les lignes à partir de la deuxième ligne et une colonne sur 2.

1.2 Matplotlib : Visualisation de données

Matplotlib est une bibliothèque qui permet de tracer des graphiques de manière simple et efficace. Elle est particulièrement utile pour la visualisation de données scientifiques et l'analyse exploratoire.

1.2.1 Fonctions utiles de Matplotlib

- **fig = plt.figure**: génère une nouvelle figure stockée dans la variable fig
- **plt.plot(x, y, label, linestyle, color)** : trace une courbe en fonction des valeurs de x et y. Arguments utiles (réutilisables dans plt.scatter et la plupart des fonctions d'affichage):
  - **x, y** : données à tracer.
  - **label** : nom de la courbe.
  - **linestyle** : style de ligne (ex: '–', '-.', ':').
  - **color** : couleur de la ligne.
- **plt.scatter(x, y, color, marker)** : trace un nuage de points.
- **plt.quiver(x, y, dx, dy)**: affiche un ensemble de flèches aux points de coordonnées x, y et de directions dx, dy (x, y, dx, dy peuvent être des tableaux pour afficher un ensemble de flèches d'un coup).
- **plt.xlabel(label)**, **plt.ylabel(label)** : ajoute des étiquettes aux axes.
- **plt.title(title)** : ajoute un titre au graphique.
- **plt.legend()** : affiche la légende.
- **plt.show()** : affiche la figure.

1.2.2 Animation en Matplotlib

La rubrique précédente détaille comment faire pour afficher une figure statique. Il est aussi possible de faire des animations, ce qui est utile dans notre cas puisque l'on modélise un phénomène dynamique. On importe la partie utile avec:

```
from matplotlib.animation import FuncAnimation
```

Pour ceux qui utilisent spyder, il est nécessaire de rentrer la ligne de code %matplotlib qt dans la console interactive (généralement située en bas à droite de la fenêtre) pour activer les animations dans spyder.

Il faut ensuite définir une fonction qui explique à Python comment mettre à jour le graphe. Dans l'exemple qui suit, j'ai détaillé les fonctions pour mettre à jour un plot, un scatter et un quiver, mais en pratique, on utilise et met à jour uniquement ce qui nous intéresse :

```
# stocker la donnee du plot dans une variable
fig = plt.figure()

lignes = plt.plot(x, y)
points = plt.scatter(x, y)
fleches = plt.quiver(x, y, dx, dy)

def animate(frame):
    # mise a jour des differents plot / scatter / quiver / etc

    lignes.set_xdata(nouveau_x) # nouvelles abcisses
    lignes.set_ydata(nouveau_y) # nouvelles ordonnees

    # nouvelles cordonnees pour les points (positions est un tableau 2D
       )
    points.set_offsets(positions)

    # nouvelles cordonnees pour les fleches (positions est un tableau 2
       D)
    fleches.set_offsets(positions)
    # nouvelles directions pour les fleches
    fleches.set_UVC(nouveau_dx, nouveau_dy)

    return

# fig indique quelle est la figure, et animate indique quelle fonction
# interval est le pas de temps entre deux images en millisecondes
anim = FuncAnimation(fig,animate,interval=2, blit=False, frames=500)
plt.show()
```

1.2.3 Ajout d'un slider pour modifier un paramètre en direct

Il est possible dans matplotlib d'ajouter un slider (curseur) pour modifier un paramètre directement depuis la figure. On importe la fonction utile avec

```
from matplotlib.widgets import Slider
```

Ensuite on doit créer un axe pour le curseur et l'initier à la valeur souhaitée, en précisant les limites et le nom:

```
    # l'argument entre [] precise la localisation et la taille du
      curseur
    ax_slider = plt.axes([0.25, 0.01, 0.5, 0.03], facecolor='
      lightgoldenrodyellow')
    # 0 et 1 sont les valeurs limites, valinit la valeur d'
      initialisation et valstep le pas
    slider_eta = Slider(ax_slider, 'Eta', 0.0, 1.0, valinit=0.2,
      valstep=0.01)  #
```

Ensuite on peut récupérer la valeur indiquée par le curseur avec

```
eta = slider_eta.val  # Mettre a jour le bruit
```

Donc en rajoutant cette ligne à la fonction animate définie plus tôt il est possible par exemple dans le modèle que l'on simule, de modifier le bruit en direct avec le curseur.

1.3 Mesure du temps

Pour mesurer des temps de calcul on peut utiliser la bibliothèque time:

```
from time import time
t = time() # sauvegarder l'instant initial
# faire des calculs
#
temps_calcul = time() - t
```

2 Définitions

- **Variables microscopiques**: positions $(x(t), y(t))$, orientations $\theta(t)$, tailles $R(t)$
- **Condition périodique :** Une condition aux limites où une particule sortant d'un bord du domaine réapparaît de l'autre côté.
- **Modèle de Vicsek :** Un modèle d'auto-organisation où des particules se déplacent à vitesse constante et interagissent pour s'aligner avec leurs voisins.
- **Paramètre d'ordre :** Une mesure de la cohérence globale du système, définie par :

$$\phi = \frac{1}{Nv_0}|\sum_i v_i| = \frac{1}{N}\sqrt{\left(\sum_{i=1}^{N}\cos(\theta_i)\right)^2 + \left(\sum_{i=1}^{N}\sin(\theta_i)\right)^2}.$$

3 Formules mathématiques

- **Distance périodique** dans une boite de taille L :

$$\Delta x_{\text{périodique}} = \Delta x - L \times \text{round}\left(\frac{\Delta x}{L}\right), \quad \Delta y_{\text{périodique}} = \Delta y - L \times \text{round}\left(\frac{\Delta y}{L}\right).$$

$$d = \sqrt{\Delta x^2_{\text{périodique}} + \Delta y^2_{\text{périodique}}}.$$

- **Mise à jour des orientations (Modèle de Vicsek) :**

$$\theta_i(t+dt) = \text{angle}\left(\sum_{j\in\text{voisins}} e^{i\theta_j(t)}\right) + \eta\xi,$$

où $\eta \in [0,1]$ est le bruit et $\xi \in [-\pi,\pi]$ est une variable aléatoire uniforme.

- **Mise à jour des positions (Modèle de Vicsek) :**

$$x_i(t+dt) = x_i(t) + v_0\cos(\theta_i(t+dt)), \quad y_i(t+dt) = y_i(t) + v_0\sin(\theta_i(t+dt)).$$