POURSUITES ET FARANDOLES

Talia LOUPIAS, Julia SALINAS, Sarah DOS SANTOS

1. INTRODUCTION

Dans le cadre de l'UE "Immersion recherche », nous avons été confrontées à un problème mathématique, et avons tenté d'y répondre.

SUJET :

« On suppose que N souris se lancent dans une course poursuite sans queue ni tête: chaque souris suit la souris devant elle et la première suit la dernière ».

Une première étape du projet consiste à **modéliser** le problème, ce qui nous conduira à l'étude d'un système d'équations différentielles d'ordre 1.
La partie principale du projet consistera alors à **résoudre numériquement** un tel système et implémenter la résolution en Python à l'aide des bibliothèques numpy et matplotlib. On réalisera alors des **simulations** en 2D et en 3D.
On pourra de plus se lancer dans **l'étude théorique** de quelques propriétés élémentaires, suggérées par exemple par les simulations réalisées, dans un cas simple.

3. THÉORIE

D'après la partie précédente, on considère la fonction :

$$\begin{aligned} f_i :\ \mathbb{R}^{2N} &\longrightarrow \mathbb{R}^2 \\ (x_0, \dots, x_{N-1}, y_0, \dots, y_{N-1}) &\longrightarrow \frac{v \times (x_{i+1} - x_i, y_{i+1} - y_i)}{[(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2]^{1/2}} \end{aligned}$$

Cette application permet d'obtenir le vecteur vitesse de la souris n°i si l'on connait les positions des N souris.

NB : on s'intéresse ici au cas en 2D, sans perte de généralités.

PROBLÈME DE DÉFINITION

Que se passe-t-il lorsque la souris n°i rattrape la souris n°$i+1$ à un instant donné ? On a alors $d_i(t) = 0$, soit un dénominateur nul. L'application ci-dessus n'est donc pas définie en un point où $x_{i+1} = x_i$ et $y_{i+1} = y_i$.

⇨ **On peut montrer que cette fonction ne se prolonge pas par continuité en un tel point :**

Pour alléger les notations, vérifions que la fonction f suivante ne se prolonge pas par continuité au voisinage de $\{(a,a,b,b) : a,b \in \mathbb{R}\}$:

$$\begin{aligned} f :\ \mathbb{R}^4 &\longrightarrow \mathbb{R} \\ (x_0, x_1, y_0, y_1) &\longrightarrow v \frac{x_1 - x_0}{[(x_1 - x_0)^2 + (y_1 - y_0)^2]^{1/2}} \end{aligned}$$

Par exemple, pour $h \in \mathbb{R}$, $f(\,(a,a,b,b) + (0,h,0,0)\,) = v\frac{h}{\sqrt{h^2}} = v\frac{h}{|h|}$ n'admet pas de limite quand $h \to 0$ puisque $\lim\limits_{\substack{h\to 0 \\ h>0}} v\frac{h}{|h|} = v > 0$ et $\lim\limits_{\substack{h\to 0 \\ h<0}} v\frac{h}{|h|} = -v < 0$

COMMENT PALLIER À CE PROBLÈME?

On pourrait :

- Modifier le problème en supposant que la souris n°i ralentit lorsqu'elle se trouve trop proche de la souris qu'elle suit.
- Ajouter un terme de lissage $\eta > 0$ au dénominateur :

$$\begin{aligned} f_i :\ \mathbb{R}^{2N}{}_i &\longrightarrow \mathbb{R}^2 \\ (x_0, \dots, x_{N-1}, y_0, \dots, y_{N-1}) &\longrightarrow \frac{v \times (x_{i+1} - x_i, y_{i+1} - y_i)}{[(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + \eta]^{1/2}} \end{aligned}$$

2. MODÉLISATION ET SIMULATIONS

MODÉLISATION : MISE EN ÉQUATIONS

Soit N le nombre de souris. On repère la souris numéro i à l'aide de ses coordonnées dans l'espace à l'instant $t \geq 0$: $X_i(t) = (x_i(t), y_i(t), z_i(t)) \in \mathbb{R}^3$.
On connait les postions initiales à $t = 0$: $X_i(0) = (x_i(0), y_i(0), z_i(0))$.
Pour $i \in \{0,1,..,N-1\}$, la souris i se déplace dans la direction de la souris $i+1$ (ou 0 si $i = N-1$) avec une vitesse constante $v > 0$.
Après un temps δt, la position de la souris est ainsi :

$$\begin{cases} x_i(t+\delta t) = x_i(t) + \delta t \times \frac{x_{i+1}(t) - x_i(t)}{d_i(t)} \times v \\ y_i(t+\delta t) = y_i(t) + \delta t \times \frac{y_{i+1}(t) - y_i(t)}{d_i(t)} \times v \\ z_i(t+\delta t) = z_i(t) + \delta t \times \frac{z_{i+1}(t) - z_i(t)}{d_i(t)} \times v \end{cases} \quad \text{où} \quad d_i(t) = \left[(x_{i+1}(t) - x_i(t))^2 + (y_{i+1}(t) - y_i(t))^2 + (z_{i+1}(t) - z_i(t))^2\right]^{1/2}$$ est la distance entre les souris i et $i+1$.

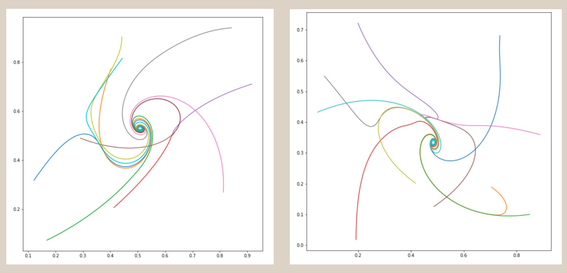
Cette version discrète permet de réaliser des simulations numériques, en mettant à jour la position des souris après δt (**cf. figure** ci-contre pour une simulation en 2D) .
Il est aussi possible de passer à un modèle continu en faisant tendre le pas de temps δt vers 0.
Le vecteur vitesse de la souris i satisfait alors les 3 équations différentielles suivantes :

$$\begin{pmatrix} x_i'(t) \\ y_i'(t) \\ z_i'(t) \end{pmatrix} = \frac{v}{d_i(t)} \begin{pmatrix} x_{i+1}(t) - x_i(t) \\ y_{i+1}(t) - y_i(t) \\ z_{i+1}(t) - z_i(t) \end{pmatrix}$$

```python
def souris_traj2D(N, t, dt,V, pos_ini = 'alea'):
    #N: nombre de souris // t: temps de simulation // dt: pas de temps // V: vitesse

    if pos_ini == 'alea': #par défaut : position initiale aléatoire
        pos = np.random.uniform(size=(N,2))
    else :
        pos = np.array(pos_ini) #pos_ini : position initiale choisie

    trajs = []
    for I in range(t): #Nombre d'itérations
        for j in range(N): #opération sur chaque position des N souris
            [x,y] = pos[j]
            if j < N-1 :
                [a,b] = pos[j+1]
                pos[j] = [x,y] + V*dt/np.sqrt((x-a)**2+(y-b)**2) * np.array([a-x ,b-y])
            else :
                [a,b] = pos[0]
                pos[j] = [x,y] + V*dt/np.sqrt((x-a)**2+(y-b)**2) * np.array([a-x ,b-y])

        trajs.append(np.copy(pos))
    return np.array(trajs)
```

RÉSOLUTION : SIMULATIONS

Exemples de trajectoires :



Figures 1 et 2:
Positions initiales aléatoires

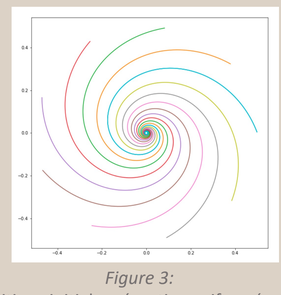


Figure 3:
Positions initiales réparties uniformément selon un cercle centré en (0,0)

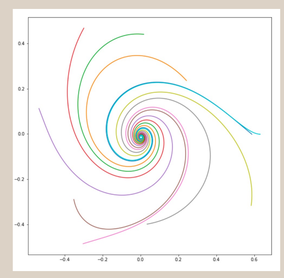
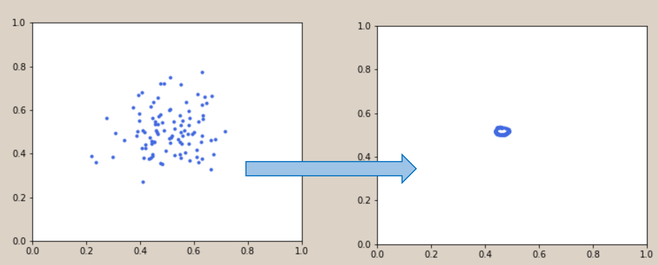


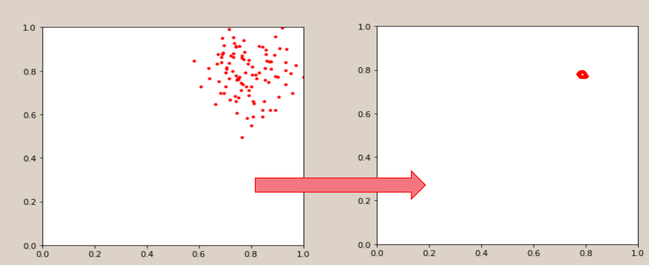
Figure 4:
Positions initiales réparties selon un cercle centré en (0,0) + perturbation

Exemples d'évolutions des positions aléatoires en 2D:



Positions initiales : selon une loi normale centrée en $(0.5\ ,\ 0.5)$

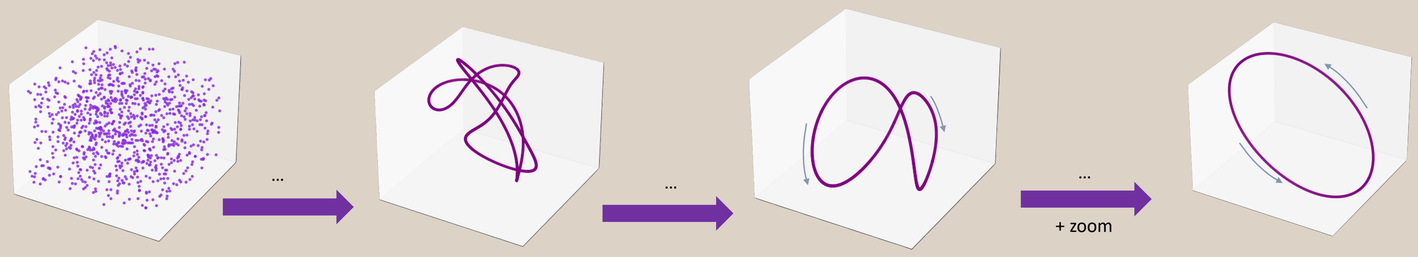
Moyenne des positions finales : (0.49243 0.48926)



Positions initiales : selon une loi normale centrée en $(0.8\ ,\ 0.8)$

Moyenne des positions finales : (0.79179 0.8048)

Exemple d'évolution des positions en 3D :



4. ANALYSE

1 – Premières observations

Tout d'abord, on remarque qu'au bout d'un certain temps de simulation, les trajectoires des souris semblent suivre une **ronde circulaire** dont le rayon tend vers 0. Cette tendance est observée indépendamment de leurs positions initiales. On conjecture alors qu'après un temps infiniment long, les positions des souris convergent vers **un unique point**, correspondant au **centre de la ronde**.
Notons que l'on retrouve toutes ces observations lorsque l'on étend notre simulation à la **3D**, et que cette ronde semble même s'inclure dans un **plan** .

2 – Influence d'une perturbation sur la position initiales

Est-ce qu'une légère perturbation des conditions initiales a un fort impact sur la trajectoire des souris ?
Après avoir fixé une position initiale pour chaque souris, on lance deux simulations : l'une avec ces conditions initiales, et une autre avec ces mêmes conditions auxquelles on ajoute un bruit gaussien. On remarque que, après un certain temps, les deux simulations sont presque similaires. Le système étudié n'est donc pas chaotique.
→ *Exemples avec un cercle bruité (figure 3 et 4):*
On voit que, lorsque le cercle initial formé par les souris est bruité, **la trajectoire des souris change peu** par rapport aux trajectoires témoins. Bien que les positions initiales soient différentes, il semble que les perturbations initiales se lissent à mesure que la simulation tourne, et que les deux trajectoires se ressemblent de plus.

3 – Influence du centrage de la distribution normale des positions initiales

Lors des deux simulations suivantes, on impose que les positions initiales des souris suivent une loi normale centré en un un point $(a,b) \in \mathbb{R}^2$.
Ainsi, après plusieurs essais, on remarque que, **si la distribution normale est centrée en** (a,b) **alors la moyenne des positions finales est environ égale au point** (a,b). On conjecture donc que, non seulement le choix des positions initiales des souris va influer sur leurs positions finales, mais également qu'il est possible de déduire la moyenne de leurs positions finales (à une petite erreur près) rien qu'en connaissant leurs positions initiales.

4 – Pour aller plus loin…

Enfin, on a remarqué que les temps de simulations étaient souvent longs. Nous aurions alors pu tenter d'optimiser notre code et réduire la complexité de l'algorithme en utilisant le module « **odeint »** plutôt que des boucles.

5. CONCLUSION

Tout au long de ce projet, nous avons eu l'opportunité de réfléchir sur un sujet mathématique complexe tout en ayant recours à diverses ressources. Nous avons donc pu développer une démarche de résolution qui est la suivante :

1- Se retrouver face à un problème ;
2- Le **modéliser**, i.e. le traduire grâce à des équations ;
3- Créer un algorithme simple pour simuler le problème et les différents paramètres ;
4- Étudier la théorie sur laquelle se fonde le sujet : déceler les différents problèmes qui peuvent se poser et **améliorer** le code en conséquence ;
5- Analyser les différentes simulations : en tirer des propriétés générales.

Lors de ces séances d'Immersion recherche, nous avons alors pu aiguiser notre logique et notre mode de raisonnement scientifique. Sans oublier que ce projet a été réalisé en trinôme et nous a donc permis de travailler en groupe, d'échanger et de collaborer ensemble afin de résoudre ce problème plus efficacement. Bien évidemment, ce projet nous a également permis d'améliorer notre maîtrise d'outils numériques tels que les notebooks Jupyter et le langage Python.

RÉFÉRENCES ET SOURCES

- Image de souris prise sur internet
- Cours oraux et écrits de Blanche BUET.

REMERCIEMENTS

Blanche BUET
Frédéric AUCHÈRE
Université Paris-Saclay

FLASHEZ CE QR CODE

Pour voir nos animations 2D et 3D

CONTACTS

julia.salinas@universite-paris-saclay.fr
sarah.dos-santos@universite-paris-saclay.fr
talia.loupias@universite-paris-saclay.fr