INFORMATIQUE
DM3 pour le 8 janvier 2024

La capacité des molécules d'ADN à participer à des mécanismes de réplication et de transcription ainsi qu'à s'organiser en chromosomes doit beaucoup à leur élasticité. Ainsi, l'étude de la réaction d'une molécule d'ADN aux contraintes mécaniques permet d'éclairer les processus biologiques mis en uvre dans une cellule vivante.

À l'aide d'une expérience et de deux modèles mécaniques d'un brin d'ADN, ce sujet propose de caractériser l'élasticité de l'ADN. Pour cela, on suppose qu'on exerce une traction sur un brin d'ADN et on cherche à établir une relation entre la force utilisée et l'allongement de la molécule.

Le seul langage de programmation autorisé dans cette épreuve est Python. Pour répondre à une question, il est possible de faire appel aux fonctions définies dans les questions précédentes. Dans tout le sujet on suppose que les bibliothèques **math**, **numpy** et **random** ont été importées grâce aux instructions

```
import math
import numpy as np
import random
```

Si les candidats font appel à des fonctions d'autres bibliothèques, ils doivent préciser les instructions d'importation correspondantes.

Ce sujet utilise la syntaxe des annotations pour préciser le type des arguments et du résultat des fonctions à écrire. Ainsi

```
def maFonction(n:int, X:[float], c:str, u) -> (int, np.ndarray):
```

signifie que la fonction **maFonction** prend quatre arguments, le premier (**n**) est un entier, le deuxième (**X**) une liste de nombres à virgule flottante, le troisième (**c**) une chaine de caractères et le type du dernier (**u**) n'est pas précisé. Cette fonction renvoie un couple dont le premier élément est un entier et le deuxième un tableau numpy. Il n'est pas demandé aux candidats de recopier les entêtes avec annotations telles qu'elles sont fournies dans ce sujet, ils peuvent utiliser des entêtes classiques. Ils veilleront cependant à décrire précisément le rôle des fonctions qu'ils définiraient eux-mêmes.

Dans ce sujet, le terme « liste »appliqué à un objet Python signifie qu'il s'agit d'une variable de type **list**. Les termes « vecteur »et « tableau »désignent des objets numpy de type **np.ndarray**, respectivement à une dimension ou de dimension quelconque. Enfin le terme « séquence »représente une suite itérable et indiąble, indépendamment de son type Python, ainsi un tuple d'entiers, une liste d'entiers et un vecteur d'entiers sont tous trois des « séquences d'entiers ».

Une attention particulière sera portée à la lisibilité, la simplicité et l'efficacité du code proposé. En particulier, l'utilisation d'identifiants significatifs, l'emploi judicieux de commentaires et la description du principe de chaque programme seront appréciés.

Une liste de fonctions utiles est fournie à la fin du sujet.

I Fonctions utilitaires

Cette partie définit quelques fonctions qui pourront avantageusement être utilisées dans la suite du sujet.

Q1. Écrire une fonction d'entête

```
def moyenne(X) -> float:
```

qui prend en paramètre une séquence de nombres et qui calcule la moyenne de ces nombres. Cette fonction ne doit pas modifier le paramètre X. Par exemple : moyenne([1, 2, 3, 4]) -> 2.5

Q2. Écrire une fonction d'entête

```
def variance(X) -> float:
```

qui calcule la variance d'une séquence de nombres, sans la modifier. Pour rappel, la variance des n nombres $x_1, x_2, \ldots, x_n$ est la moyenne des carrés des écarts à la moyenne, c'est-à-dire

$$\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2 = \frac{1}{n}\sum_{i=1}^{n}x_i^2 - \overline{x}^2 \text{ avec } \overline{x} = \frac{1}{n}\sum_{i=1}^{n}x_i$$

Par exemple : `variance([1, 2, 3, 4]) -> 1.25`

Q3. Écrire une fonction d'entête

```
def somme(M):
```

qui prend en paramètre une séquence imbriquée, de profondeur et de structure quelconques, dont tous les composants élémentaires sont des nombres, et calcule la somme de tous ces éléments.
Par exemple : `somme([[[1, 2], [3, 4, 5]], 6, [7, 8], 9]) -> 45`

Indication - *L'expression booléenne isinstance(x, numbers.Real) permet de tester si x est un scalaire numérique. Par exemple*

```
isinstance(1, numbers.Real) -> True
isinstance(2.3e4, numbers.Real) -> True
isinstance([1, 2, 3], numbers.Real) -> False
```

II Mesures expérimentales

Depuis quelques décennies, des équipes de recherche réussissent à isoler un brin d'ADN et à mesurer ses propriétés mécaniques. Cette partie s'appuie sur une série d'expériences réalisées dans les années 1990, en particulier au laboratoire de physique statistique de l'École Normale Supérieure.

Une molécule d'ADN est attachée à une de ses extrémités sur un support transparent, une microbille magnétique de diamètre 2,5 μm est greffée à son autre extrémité. À l'aide d'aimants, la molécule d'ADN est soumise à une force de traction notée $\overrightarrow{F}$.

Afin de caractériser l'élasticité du brin d'ADN, on cherche à mesurer son allongement pour différentes intensités de la force de traction.

L'intensité de la force de traction n'est pas accessible directement, nous allons l'évaluer indirectement. Une fois le brin d'ADN mis en tension, son extrémité matérialisée par la bille ne reste pas immobile, elle est animée d'un mouvement aléatoire, dit mouvement brownien, dû à l'agitation des molécules du liquide qui l'entourent. En assimilant la molécule à un ressort, on montre que l'intensité de la force de traction est inversement proportionnelle aux fluctuations quadratiques moyennes de la position de la bille.

Une caméra CCD reliée à un ordinateur permet de photographier l'image de la bille (figure 1). Compte tenu de la taille de cette bille, on obtient une image de diffraction que nous allons analyser pour déterminer la position de la bille.

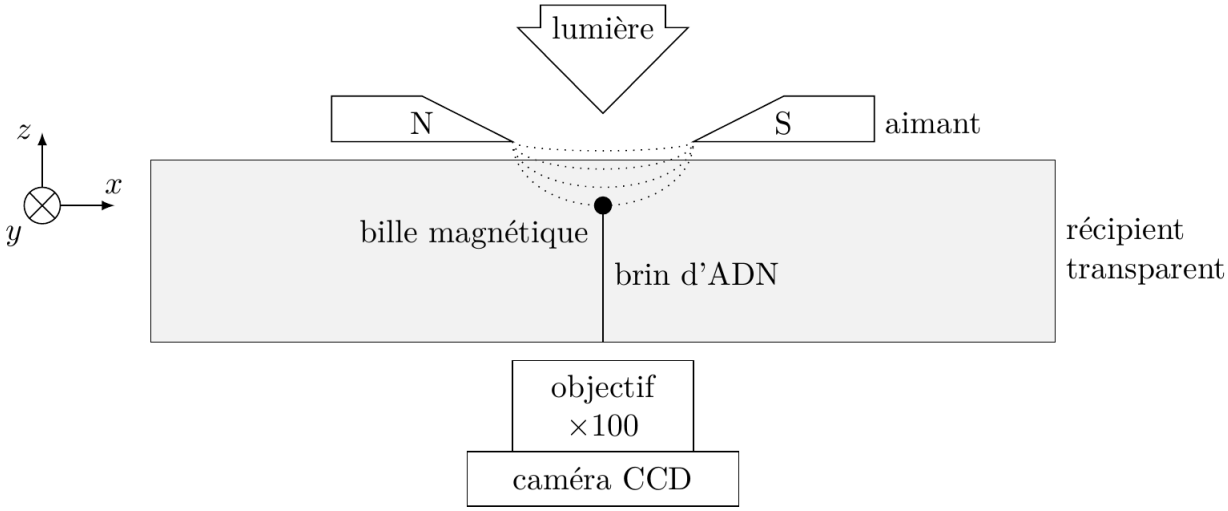

Figure 1 - Schéma du dispositif expérimental (échelle non respectée)

II.A. Position de la bille

La figure 2 donne, à gauche, un exemple d'image obtenue par la caméra CCD. Cette caméra est pilotée par un programme Python qui récupère chaque image sous la forme d'un tableau d'entiers à deux dimensions. Les images obtenues sont en niveau de gris, chaque pixel est codé sur 8 bits, soit une valeur comprise entre 0 (noir) et 255 (blanc).

Afin de repérer le centre de la figure de diffraction, l'image est convertie en noir et blanc inversé suivant une valeur seuil du niveau de gris : les pixels au-dessus du seuil deviennent noirs, ceux en dessous deviennent blancs.

Une fois ce seuillage effectué, on calcule le barycentre des pixels blancs de l'image seuillée pour obtenir la position de la bille.

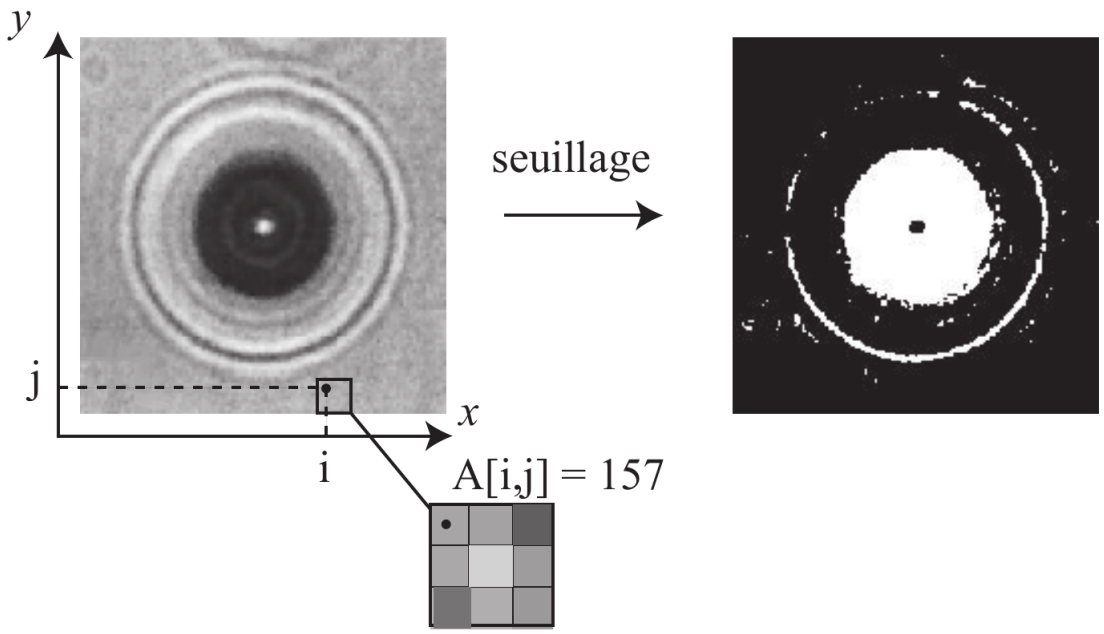

Figure 2 - Figure de diffraction d'une bille et opération de seuillage

On rappelle que l'abscisse (respectivement ordonnée) du barycentre d'un ensemble de points de même poids est la moyenne des abscisses (respectivement ordonnées) des points considérés.

Q4. Écrire une fonction d'entête

```
def seuillage(A:np.ndarray, seuil:int) -> np.ndarray:
```

qui prend en paramètre un tableau d'entiers à deux dimensions représentant un cliché de la caméra CCD et construit un tableau de même forme contenant la valeur 1 là où la valeur des pixels de l'image originale est strictement inférieure au seuil et la valeur 0 ailleurs (pixels supérieurs ou égaux au seuil).

Q5. Écrire une fonction d'entête

```
def pixel_centre_bille(A:np.ndarray) -> (int, int):
```

qui prend en paramètre l'image seuillée telle que produite par la fonction seuillage et renvoie les indices (ligne et colonne) du pixel le plus proche du centre de la bille (barycentre des pixels à 1).

On dispose de la fonction d'entête

```
def prendre_photo() -> np.ndarray:
```

qui déclenche la prise d'un cliché par la caméra CCD et renvoie l'image prise sous la forme d'un tableau à deux dimensions tel que décrit plus haut.

Q6. Écrire une fonction d'entête

```
def positions(n:int, seuil:int) -> [(int, int)]:
```

qui prend `n` photographies de la bille et renvoie la liste de ses positions dans chaque photographie en seuillant les images à la valeur seuil.
Le résultat de cette fonction est donc une liste de `n` couples de deux entiers correspondants à l'indice de ligne et de colonne des positions successives du centre de la bille au cours de son mouvement brownien.

Le capteur CCD est positionné parallèlement au plan (xOy) et ses pixels sont carrés. La caméra a été calibrée dans les conditions de l'expérience : un pixel correspond à un carré du plan (xOy) de côté t.

Q7. Définir une fonction d'entête

```
def fluctuations(P:[(int, int)], t:float) -> float:
```

qui prend en paramètre une liste de positions successives de la bille (telle que produite par la fonction positions) et la longueur correspondant à un pixel et calcule la valeur moyenne des déplacements quadratiques de la bille :
moyenne des carrés des écarts entre chaque position mesurée et la position d'équilibre de la bille (correspondant au barycentre des différentes positions observées).