Jean-Eudes BORNERT

Tom CHAUVEL

ECOLE SUPERIEURE
D'INGENIEURS DE RENNES

TP 1 – Probabilités

Université de Rennes

CUPGE 2 ESIR

Année universitaire 2022 - 2023

Introduction

Ce travail consiste à simuler l'expérience aléatoire dite de l'aiguille de Buffon en utilisant les méthodes de génération de nombres aléatoires de la librairie Numpy, sous Python. Avant cela, nous allons étudier certaines fonctionnalités de Python qui seront ensuite utilisées.

Partie A : Prise en main de Python

Etape 1 :

Après avoir installé Anaconda, nous allons utiliser l'éditeur de code Spyder.

Etape 2 :

Après avoir créé un fichier, nous importons la librairie Numpy.

Etape 3 :

Nous voulons générer un vecteur x contenant N=100 réalisations indépendantes d'une variable aléatoire gaussienne de paramètres m=0 et σ=1 (m désigne la moyenne et σ l'écart-type).

Pout cela nous utilisons le script suivant (figure 1) :

```
import numpy as np
import numpy.random as rd

x = rd.normal(0,1,100)
print(x)

>>> [-2.00384165e-02 , … , -1.21511656e+00]
```

Figure 1 : Script utilisé pour générer le vecteur x affiché

Ce script a généré une matrice de dimension 1 x 100 suivant la loi normale avec une moyenne de 0 et un écart-type de 1.

Maintenant modifions cette partie de code pour avoir une matrice carrée de dimension (5,5) avec pour moyenne m=10 et un écart type σ =0,1 (figure 2).

```
import numpy as np
import numpy.random as rd

mat = rd.normal(10,0.1,(5,5))
print(mat.size,mat.shape)

>>> 25 (5, 5)
```

Figure 2 : Script utilisé pour générer la matrice carrée

On obtient une matrice de dimension (5,5) ce qui est confirmé par *mat.shape* qui permet de connaitre le nombre de colonnes et le nombre de lignes. Sa taille est de 25, ce qui logique puisque l'on a 5*5=25 éléments dans notre matrice, donné grâce à *mat.size* qui calcule le nombre d'éléments dans une matrice.

Etape 4 :

Tout d'abord nous importons matplotlib qui permet un affichage graphique. Nous souhaitons afficher ici la représentation graphique des N réalisations contenues dans x. La figure est affichée grâce à la commande plt.plot() (ligne 6 figure 3). Les lignes 7, 8 et 9 servent à habiller le graphique. Nous obtenons le graphique affiché dans la figure 3 :

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

x = rd.normal(0,1,100)

plt.figure()
plt.plot(x)
plt.title('Représentation du vecteur aléatoire x')
plt.xlabel('Indice du vecteur')
plt.ylabel('Valeur')
plt.show()

>>>
```

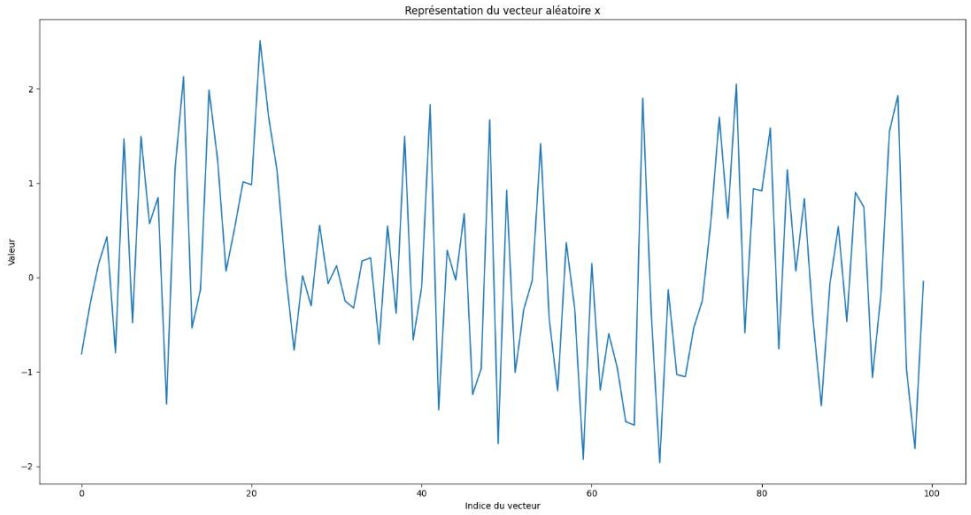

Figure 3 : Script et graphique permettant de représenter le vecteur aléatoire x

On obtient donc un ensemble de points situés aux alentours de la moyenne prédéfinie qui est de 0. En omettant aussi les points les plus extrêmes (minoritaires dans l'échantillons) on observe bien que la distance entre la moyenne et ces points est comprise entre 0 et 1. Ces points extrêmes sont là pour compenser les points plus proches de 0 que de 1 pour contenir l'écart type à 1.

Etape 5 :

A cette étape nous allons construire l'histogramme correspondant aux valeurs du vecteur aléatoire x. Nous utilisons la commande plt.hist (ligne 7 figure 4). Nous choisissons de regrouper nos valeurs dans

20 colonnes (M=20) en faisant la somme des valeurs comprises dans un intervalle. Nous obtenons la figure 4 suivante :

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

x = rd.normal(0,1,100)

M=20
plt.figure()
plt.hist(x,M)
plt.show()

>>>
```

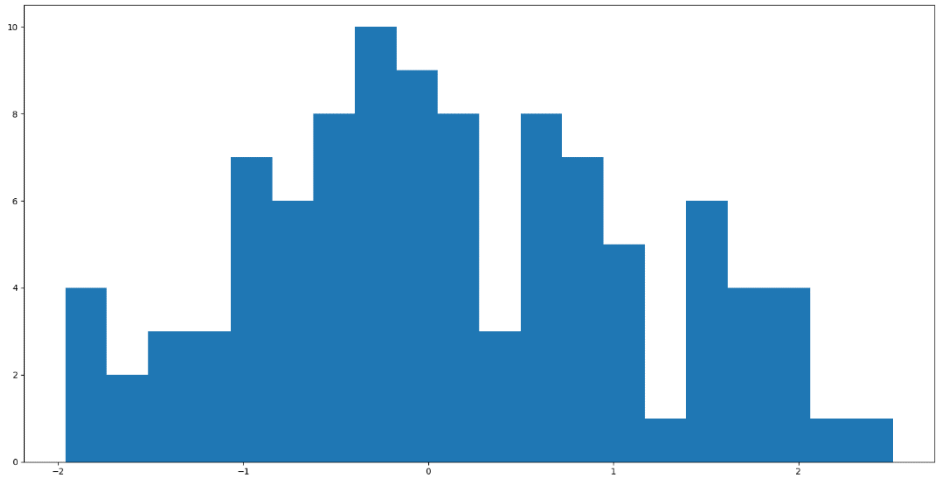

Figure 4 : Script et histogramme permettant de représenter le vecteur aléatoire x à l'aide de 20 colonnes

Nous modifions maintenant le script précédent en choisissant non plus de scinder l'ensemble en 20 parts égales, mais en regroupant les valeurs selon un pas de 0,1 entre les valeurs –3 et 3 exclu grâce à la commande *np.arange*. Nous utilisons le script de la figure 5 et nous obtenons un nouvel histogramme.

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

x = rd.normal(0,1,100)

pas = 0.1
bins = np.arange(-3,3,pas)
plt.figure()
plt.hist(x,bins)
plt.show()

>>>
```

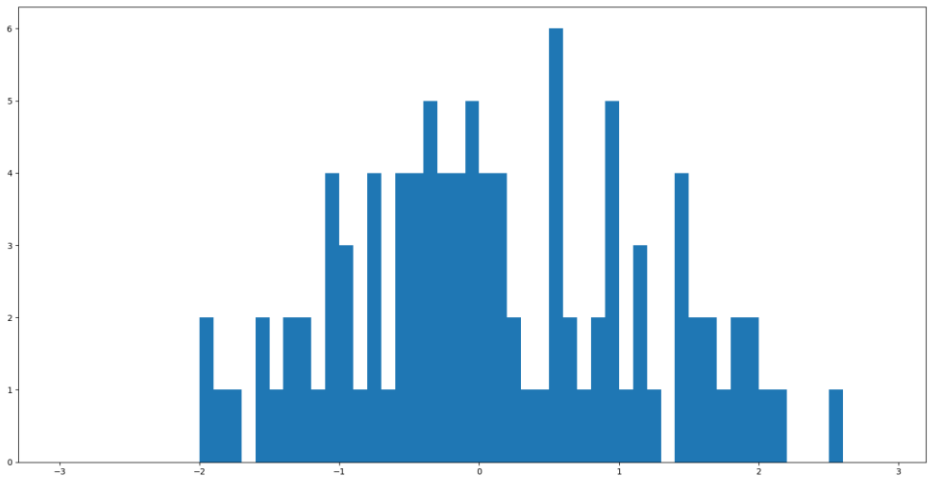

Figure 5 : Script et histogramme correspondant pour le vecteur x

Dans les deux cas on observe, avec un peu d'imagination, un semblant de courbe gaussienne suivant une loi normale. En faisant varier M, on observe que des valeurs de M trop faibles rendent l'histogramme peu lisible car le regroupement des valeurs de x est trop grossier ; avec des valeurs de M très élevées, l'histogramme est là encore peu lisible par manque de regroupement des valeurs. Il faut donc trouver une valeur de M optimale par rapport à la dispersion des données. Le même constat peut être fait en modifiant le pas.

Etape 6 :

Nous voulons à cette étape représenter graphiquement la courbe associée à la densité de probabilité correspondant au vecteur x. Pour cela nous disposons de deux méthodes. Nous devons tout d'abord créer le vecteur x_px à l'aide de la fonction *arange* de numpy, comme vu précédemment. La première méthode consiste à utiliser une boucle for et la seconde à utiliser une méthode vectorisée. Les deux méthodes permettent de calculer les valeurs d'ordonnée (figure 6) mais la méthode vectorisée est la plus rapide.

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

x = rd.normal(0,1,100)

m = 0
sigma = 1
pas = 0.05
x_px = np.arange(-3,3,pas)

Méthode vectorisée :
px = 1 / (sigma * np.sqrt(2*np.pi)) * np.exp(-1/(2*sigma**2)*(x_px-m)**2)
plt.plot(x_px,px)
plt.show()

Méthode boucle for :
px2 = np.zeros(px.size)
for i in range(px.size) :
    px2[i] = 1 / (sigma * np.sqrt(2*np.pi)) * np.exp(-1/(2*sigma**2)*(x_px[i]-m)**2)
plt.plot(x_px,px2)
plt.show()


>>>
```

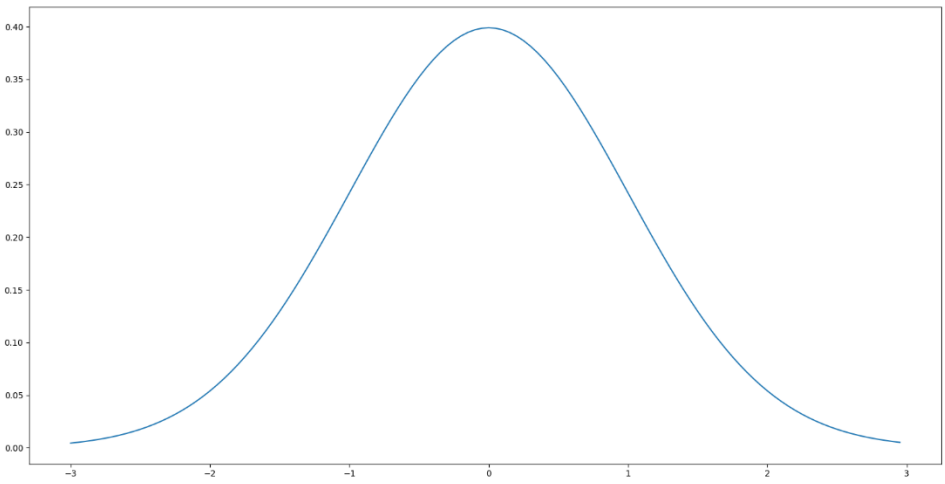

Figure 6 : Script utilisé pour représenter graphiquement la courbe associée à la densité de probabilité du vecteur x

En ajoutant "*density=true*" dans les appels de commande des histogrammes, on normalise les données, c'est-à-dire qu'au lieu d'avoir des résultats très grands, on les ramène à des valeurs plus analysables, comprises entre 0 et 1.

En appliquant l'une comme l'autre des deux méthodes, on obtient une parfaite courbe gaussienne ce qui correspond bien à la variable aléatoire gaussienne définie initialement.

En augmentant le nombre N à 1000, on augmente le nombre de valeurs prises par x ce qui permet d'obtenir des histogrammes dont l'allure s'apparente plus à celle d'une courbe gaussienne (figure 7).

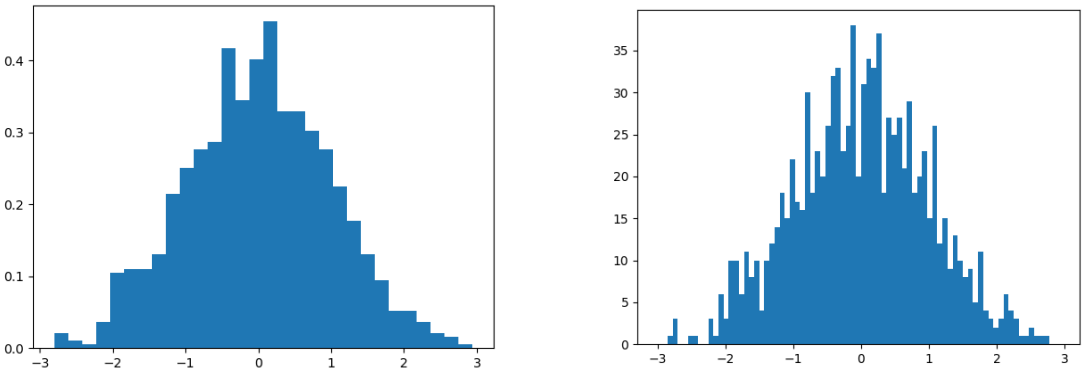

Figure 7 : Histogrammes obtenus pour N=1000 en utilisant les mêmes scripts que précédemment :
- A gauche : M=20
- A droite : pas de 0,1

Partie B : Aiguille de Buffon

Question 1

Dans cette partie, nous devons réaliser plusieurs lancers successifs d'une aiguille sur une grille quadrillée. Notre but est de déterminer expérimentalement la probabilité que l'aiguille touche une ligne horizontale. Cela viendra confirmer ou non la théorie établie par Buffon qui dit que cette probabilité est :

$$P(E) = \frac{2}{\pi} \times \frac{L}{a}$$

Avec L la longueur de l'aiguille et a la distance entre deux lignes horizontales. Ici, comme L=a,

$$P(E) = \frac{2}{\pi}$$

Le script présent en annexe (ex_2_q1.py) est utilisé pour simuler cette expérience aléatoire et la visualiser géométriquement.

On obtient les résultats suivants (figure 8) :

```
>>>
```

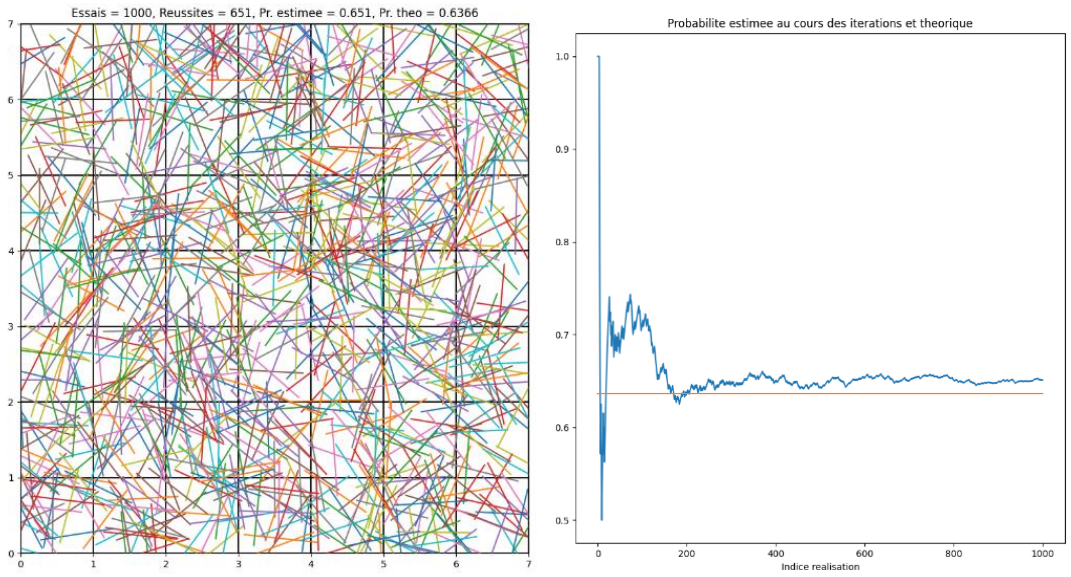

Figure 8 : Résultats obtenus à l'aide du script :
- A gauche : différentes positions de l'aiguille pour les différents essais ;
- A droite : En bleu : Probabilité estimée au cours des itérations, en orange : probabilité théorique calculée.

En comparant les valeurs expérimentales et la valeur théorique, on observe que la probabilité expérimentale tend vers la valeur théorique calculée lorsque le nombre d'essais tend vers l'infini. Au bout de 1000 essais, la différence entre la valeur théorique et la valeur expérimentale est de 2%. Ceci vient confirmer la véracité de la formule de Buffon.

Question 2

Ici nous faisons la même chose qu'auparavant, sauf que l'on rajoute des lignes verticales espacées non pas d'une distance a comme précédemment, mais d'une distance 2a .

La probabilité théorique se calcule de la manière suivante :

Si l'on considère les événements suivants :

- A : le bâton touche une ligne horizontale : $P(A) = \frac{2}{\pi} \times \frac{L}{a}$;
- B : le bâton touche une ligne verticale : $P(B) = \frac{2}{\pi} \times \frac{L}{2a}$

On obtient :

$$P(E) = P(A) + P(B) - P(A \cap B)$$

Donc :

$$P(E) = \frac{2}{\pi} \times \frac{L}{a} + \frac{2}{\pi} \times \frac{L}{2a} - \frac{2}{\pi} \times \frac{L}{2a} \times \frac{2}{\pi} \times \frac{L}{a}$$

Ici, comme L=a,

$$P(E) = \frac{3\pi - 2}{\pi^2}$$

Le script présent en annexe (exo_2_q2.py) est utilisé pour calculer P(E) et la visualiser graphiquement (figure 9) :

```
>>>
```

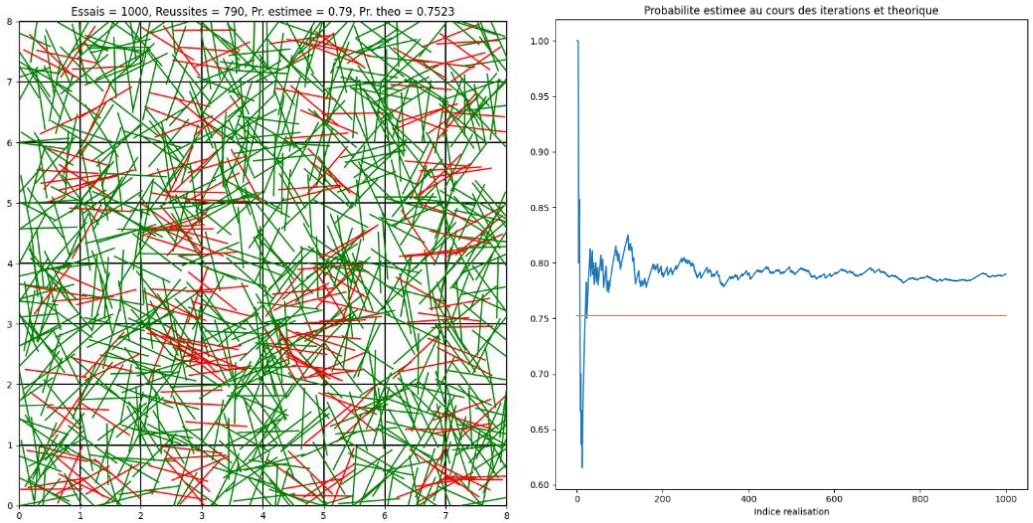

Figure 9 : Résultats obtenus à l'aide du script :
- A gauche : différentes positions de l'aiguille pour les différents essais ;
- A droite : En bleu : Probabilité estimée au cours des itérations, en orange : probabilité théorique calculée.

On observe que la probabilité P(E) dans cette expérience est supérieure à celle du premier essai. Comme précédemment, la probabilité estimée tend après un nombre important d'essais vers une valeur proche de la valeur calculée (avec cependant une erreur de 5%). On obtient donc dans ce cas des résultats expérimentaux moins bons que dans le cas précédent, ce qui s'explique par le cumul des erreurs associées à P(A) et à P(B).

Question 3

Maintenant remplaçons les aiguilles par des anneaux de diamètre L et de coordonnée $x = \frac{N \times a}{2}$.

Dans la mesure où le diamètre d'un anneau est égal à la distance entre les lignes, la probabilité qu'un anneau touche une ligne est de 1.

On utilise le script présent en annexe (exo_2_q3.py) pour simuler cette expérience aléatoire et la visualiser géométriquement (figure 10) :

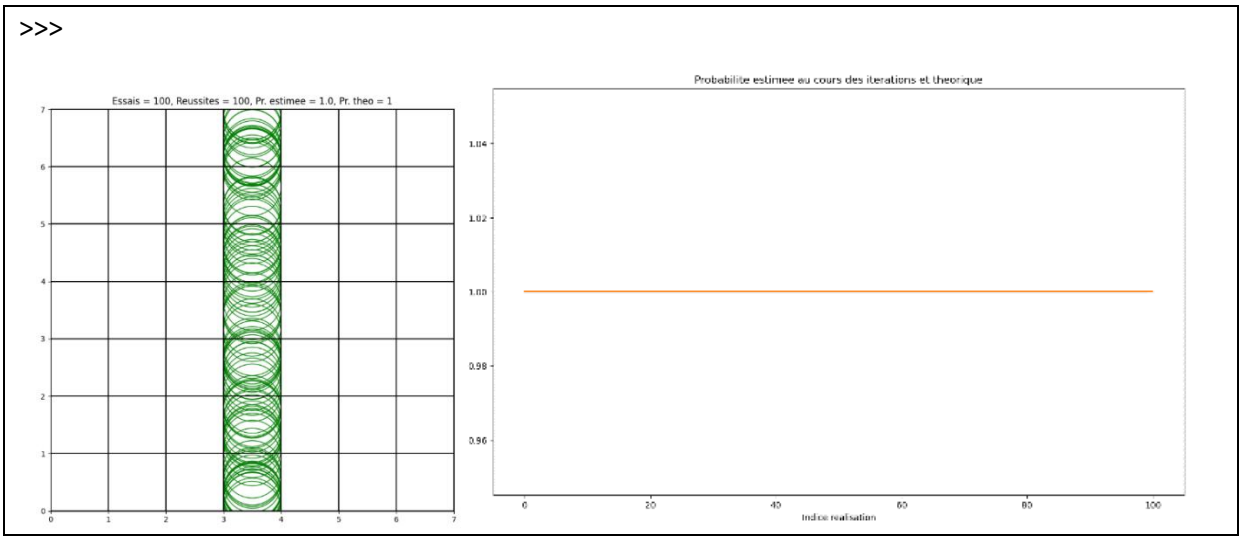

Figure 10 : Résultats obtenus à l'aide du script :
- A gauche : différentes positions de l'aiguille pour les différents essais ;
- A droite : En bleu : Probabilité estimée au cours des itérations, en orange : probabilité théorique calculée (superposées)

On obtient bien un résultat cohérent du fait que, l'anneau étant de même taille que la distance entre deux lignes, touchera forcément une ligne verticale. En comparant ces résultats avec ceux obtenus à la question 1, on observe que :

$$P\left(E_{cercle}\right) = P\left(E_{aiguille}\right) \times \frac{\pi}{2}$$

Cette formule proposée pour le cas particulier étudié ici peut être facilement expliquée dans la mesure où l'expérience utilisant des cercles revient à une série d'expériences utilisant des aiguilles et le calcul de probabilité revient à faire la somme des probabilités pour $\frac{\pi}{2}$ aiguilles « tournant » autour du centre M de l'anneau, selon le schéma présent sur la figure 11.

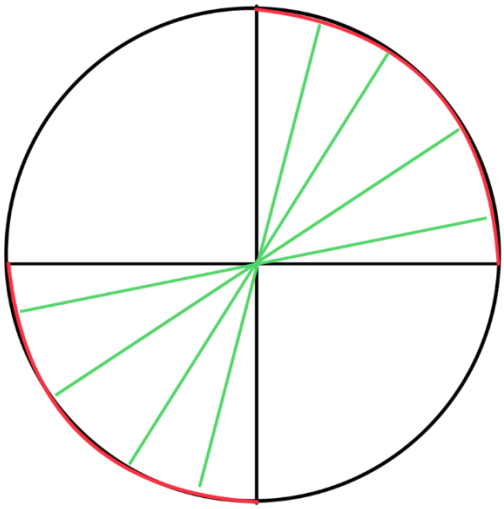

Figure 11 : Schéma représentant l'anneau en noir ainsi que quelques bâtons en vert

Conclusion

Dans ce TP, nous avons pu nous familiariser avec les outils mathématiques de Numpy ainsi que certaines fonctions de Python destinées à simuler des expériences aléatoires. Dans un second temps, nous avons utilisé ces outils pour simuler l'expérience aléatoire dite de l'aiguille de Buffon. Cela nous a permis de vérifier la formule proposée par Buffon donnant la probabilité associée à cette expérience. Nous avons ensuite étudié une variante de l'expérience précédente consistant à remplacer l'aiguille par un anneau et déduit une relation entre les probabilités mesurées dans les deux types d'expériences.