Jean Estadieu
Gael Bellanger

Groupe C

Dossier Marche aléatoire

Sommaire

Introduction 3
Partie I : Le projet 4
Partie II : Les résultats 6
Partie III : La Méthodes de travail 10
Conclusion 11

Introduction

Préambule :

Nous sommes deux étudiants en seconde année du DUT informatique à l'IUT Paul Sabatier de Toulouse. Dans le cadre de nos études, on se propose d'étudier des marches aléatoires sur différents espaces. Le langage Python va nous permettre d'observer leurs comportements en temps long. Pour la compréhension de notre dossier, on a jugé nécessaire de rappeler quelques définitions importantes que nous allons réutiliser tout au long du dossier.

Définition :

Marche aléatoire : est un modèle mathématique d'un système possédant une dynamique discrète composée d'une succession de pas aléatoires, ou effectués au hasard.

Récursive : Une suite est récursive si pour tout n entier, il existe m>n tel que Um = 0.

Transitive : une relation transitive est une relation pour laquelle une suite d'objets reliés consécutivement aboutit à une relation entre le premier et le dernier.

Choix du sujet :

Concernant le choix du sujet nous avons pris la marche aléatoire car elle est utilisée dans de nombreux domaines comme l'économie ou l'informatique.

Les objectifs du projet :

En partant d'un modèle simple, comme le jeu du pile ou face, nous introduirons les notions de transitivité et de récursivité pour ensuite complexifier le jeu et analyser les résultats obtenus.

Jean Estadieu
Gael Bellanger

Partie I : Le projet

Ce que l'on a fait :

Le projet a été divisé en 5 étapes. A chaque étape on a créé un nouveau fichier python avec comme base le fichier précédent. Cela nous a permis de conserver toute l'évolution du projet et d'apporter différentes modifications aux codes.

Première étape :

Pour commencer, nous avons écrit un algorithme en Python qui permet de simuler les n premiers éléments d'une marche aléatoire de paramètre p. Pour ce faire, on demande à l'utilisateur de rentrer la valeur du paramètre entre 0 et 1 et les nombres n d'éléments à simuler. A chaque fin de marche, le programme doit nous retourner la valeur maximale et minimale qu'elle a pu prendre.

Seconde étape :

Par la suite nous avons complexifié un peu le code. Cette fois-ci, le code simulera les N marches aléatoires avec comme paramètre p et comme nombres de lancers n. Ces variables sont choisies par l'utilisateur dans la console. Le code produit un graphique affichant les résultats pour pouvoir les comprendre et les analyser.

Troisième étape :

Même code que précédemment auquel on ajoute tous les temps de retour en 0, nous avons rajouté un compteur qui compte le nombre de retours en 0 pour chaque marche et qui affiche dans un diagramme en bâtons le nombre de retours en 0 à chaque temps donné.

Quatrième étape :

Passage en **2 dimensions**, avec p=q=r=s=1/4.

Ici le code consiste à simuler la marche d'un homme ivre dans une ville, l'homme choisit une rue au hasard et il peut aussi revenir sur ses pas. Le code produit les mêmes graphiques que précédemment.

Cinquième étape :

Passage en **3 dimensions** avec cette fois-ci une probabilité de 1/6.

Dans ce cas-là, on simule plutôt un oiseau qui se déplace dans un plan en 3 dimensions.

Ces deux dernières étapes nous ont dirigés vers le théorème de Polya que nous verrons par la suite.

Ce que l'on a compris :

Le fait d'avoir commencé par la simulation d'un jeu simple (le jeu du pile ou face) et d'avoir ensuite rendu la chose beaucoup plus complexe, en y introduisant des notions encore jamais vues, a été enrichissant pour nous.

Effectuer des tests avec différentes valeurs et différents paramètres, nous a permis de comprendre nos résultats et de pouvoir les utiliser pour en tirer des conclusions.

Les difficultés rencontrées :

Pour arriver au bout du projet, nous avons dû effectuer beaucoup de recherches pour comprendre ce qu'on attendait exactement de nous. Par exemple le théorème de Polya nous a posé quelques difficultés de compréhension que l'on a su résoudre grâce à nos recherches.

L'une des plus grandes difficultés qu'on a rencontrées a été d'apprendre le Python. En effet nous avions fait un peu de Python lors de la première année, mais nous n'avions pas toutes les bases pour réaliser ce projet. On a donc cherché sur des forums, posé des questions pour pouvoir se familiariser avec ce langage.

Les sources utilisées :

- https://fr.wikipedia.org/wiki/Marche_al%C3%A9atoire
- https://secouchermoinsbete.fr/79494-avec-le-theoreme-de-polya-l-homme-ivre-rentre-chez-lui
- https://www.bibmath.net/dico/index.php?action=affiche&quoi=./m/marchealea.html
- https://www.techno-science.net/glossaire-definition/Marche-aleatoire-page-3.html

Partie II : Les résultats

Marche aléatoire à une dimension :

P = 0.5 et n = 20

```
Entrer le nombres de lancers :20
le resultat final avec comme paramètre :
0.5
est :
-6
le maximum est :
1
le minimum est :
-6
```

```
Entrer le nombres de lancers :20
le resultat final avec comme paramètre :
0.5
est :
12
le maximum est :
12
le minimum est :
-1
```

Après plusieurs tests effectués avec 20 comme nombres de lancers on constate que si le paramètre est 0.5 la marche est équilibrée et donc récursive.

P = 0.9 et n = 20

```
Entrer le nombres de lancers :20
le resultat final avec comme paramètre :
0.9
est :
18
le maximum est :
18
le minimum est :
0
```

```
Entrer le nombres de lancers :20
le resultat final avec comme paramètre :
0.9
est :
20
le maximum est :
20
le minimum est :
0
```

P = 0.6 et n = 10 000

```
Entrer le nombres de lancers :10000
le resultat final avec comme paramètre :
0.6
est :
2076
le maximum est :
2083
le minimum est :
-2
```

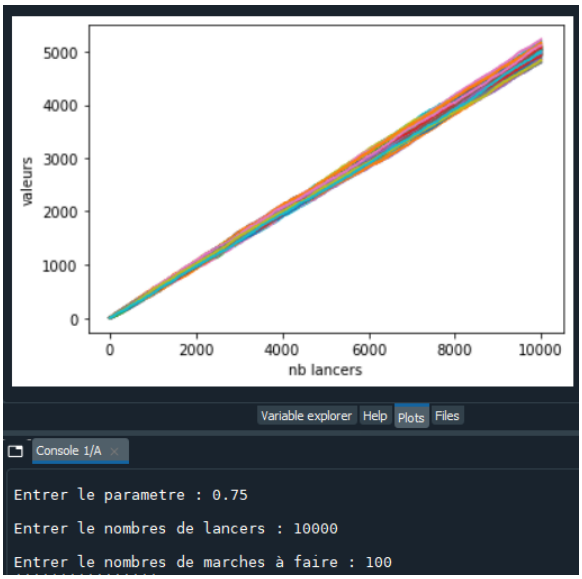
```
Entrer le nombres de lancers :10000
le resultat final avec comme paramètre :
0.6
est :
1990
le maximum est :
1990
le minimum est :
-1
```

Si le paramètre p change et que la marche n'est plus équilibrée, elle devient transitive. Dans notre cas on remarque bien que même en prenant p = 0.6, la marche aléatoire va tendre vers l'infini.

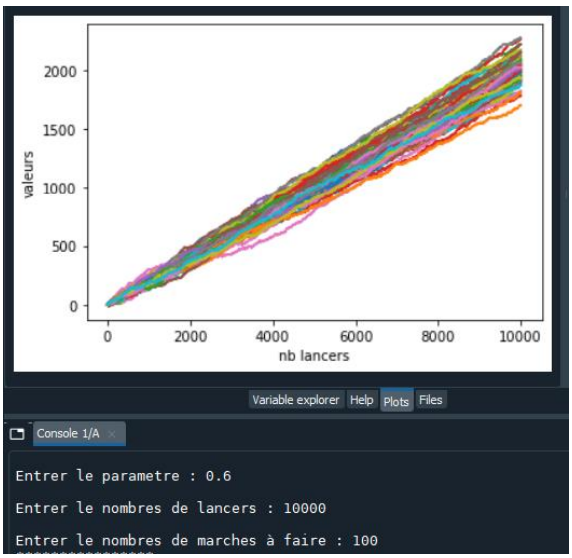
Jean Estadieu
Gael Bellanger

Groupe C

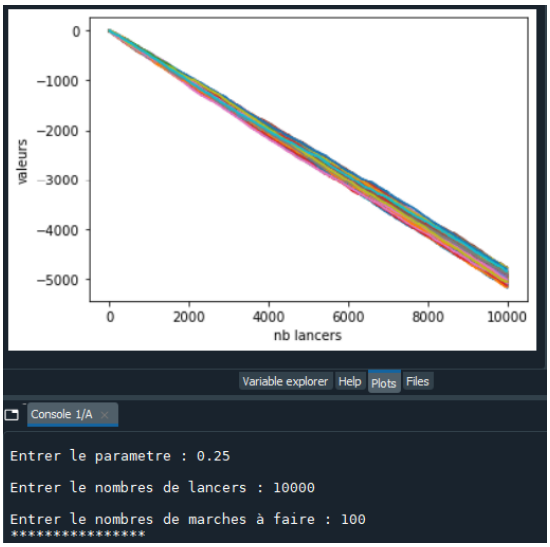
N = 100 ; p = 0.75 ; n = 10000



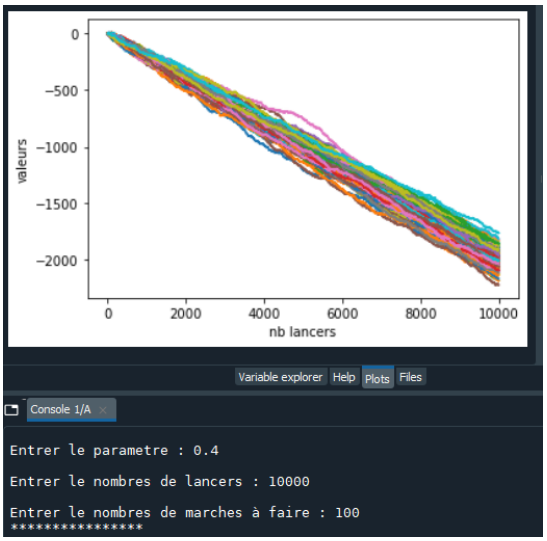
N = 100 ; p = 0.6 ; n = 10000



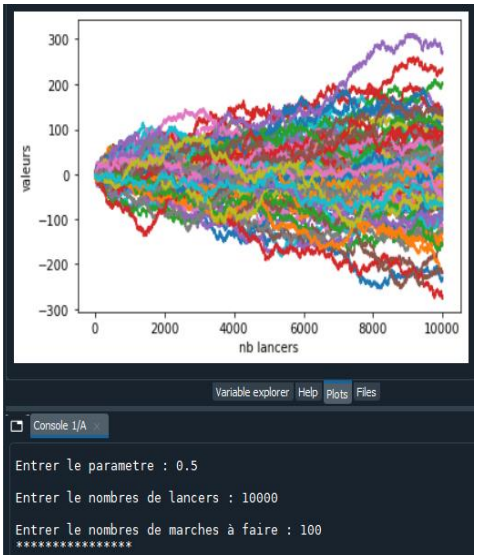
N = 100 ; p = 0.25 ; n = 10000



N = 100 ; p = 0.4 ; n = 10000



N= 100 ; p= 0.5 ; n = 10000



Dans ces différentes situations, on peut clairement voir que lorsque la marche est déséquilibrée ($p \neq 0.5$), les marches semblent être transitives.

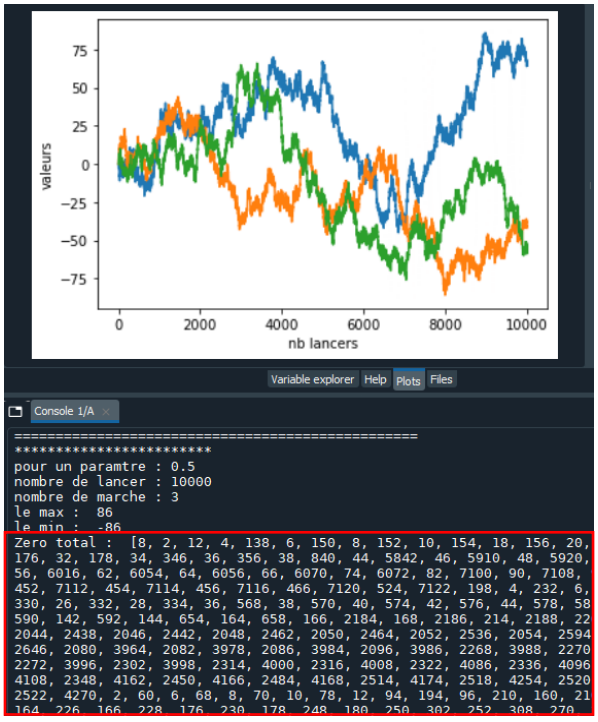
En effet si :

- p>0.5 Les marches aléatoires vont tendre vers $+\infty$
- P<0.5 Les marches aléatoires vont tendre vers $-\infty$
- P = 0 La marche est équilibrée et récursive

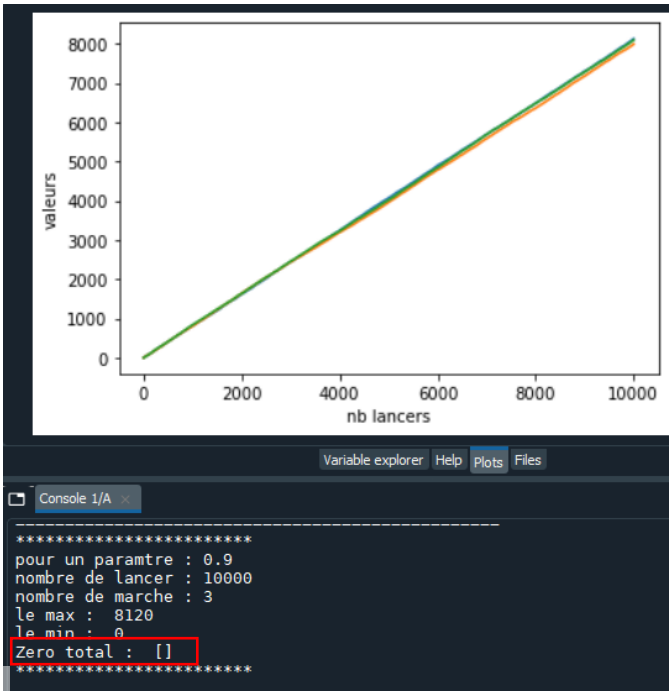
Jean Estadieu
Gael Bellanger

Groupe C

N = 3 ; p = 0.5 ; n = 10000

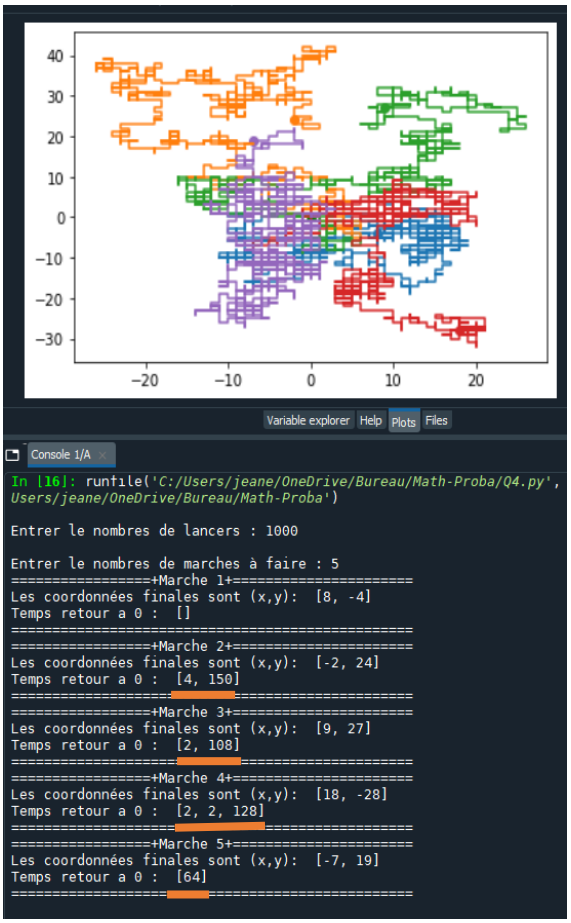


N = 3 ; p = 0.9 ; n = 10000



En rajoutant à cet algorithme les temps de retour en 0, on remarque que les marches déséquilibrées ont moins de chance de retourner en 0 qu'une marche équilibrée.

Marche aléatoire à deux dimensions :



Dans le cadre de la 2 dimensions, on se place dans un processus qui modélise la marche d'un homme ivre (4 directions possibles)

P=q=r=s= ¼

Ici nous avons simulé 5 marches aléatoires ; l'homme commence son parcours à l'origine au point de coordonnées [0,0], le code nous retourne pour chaque marche le point de coordonnées à la fin des 1000 choix de rue effectués par l'homme. On remarque que sur 4 des 5 marches effectuées, l'homme est retourné au moins une foi à l'origine

On peut donc en conclure qu'une marche aléatoire équilibrée est récursive pour les dimensions 1 et 2

Jean Estadieu
Gael Bellanger

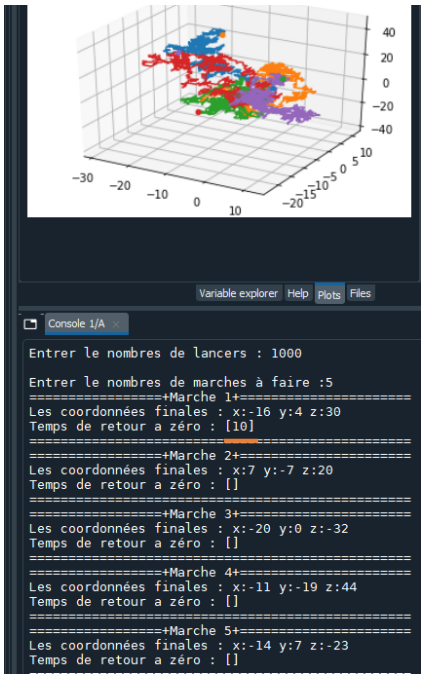
Groupe C

Marche aléatoire à trois dimensions :

On se propose ici d'étudier les marches aléatoires en 3 dimensions. On ne s'intéresse qu'au cas équilibré avec comme probabilité 1/6 :

On effectue exactement le même test que pour les deux dimensions, c'est-à-dire 5 marches aléatoires avec 1000 pas.

A chaque pas, il y a six mouvements possibles : en avant, en arrière, à droite, à gauche, en haut, en bas.



Dans ce cas-là, on remarque que seule la première marche a un temps de retour en 0.

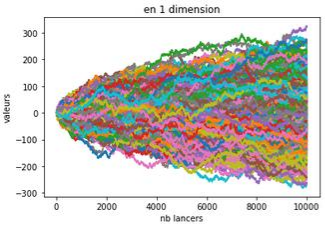
Après avoir effectué ce test plusieurs fois d'affilées, puis en changeant les valeurs du nombre de lancers, on s'est rendu compte que pour les dimensions 3 et au-dessus, les marches aléatoires ne sont plus récursives.

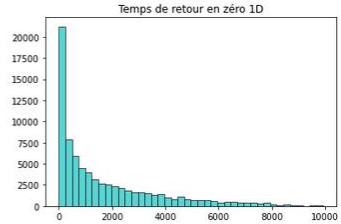
Il s'agit là du théorème de Polya, lorsque chaque déplacement est équilibré pour savoir si la marche aléatoire est récursive ou transitive, il faut regarder sa dimension.

Bonus :

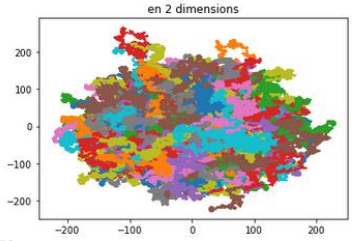
On a rajouté un compteur à nos dimensions 2D et 3D qui permet de compter le nombre de retours en 0 et d'afficher dans un diagramme en bâton à quel moment les marches sont retournées en 0.

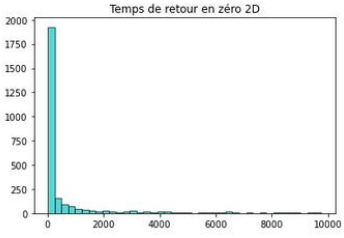
```
nombre de lancer : 10000
nombre de marche : 1000
le max :  327
le min :  -283
il y a eu 76047 retour en zéro dans toutes les marches
**********
```



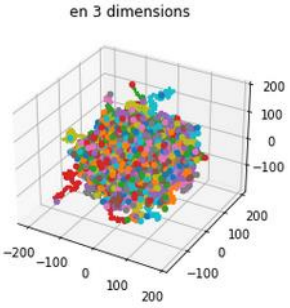


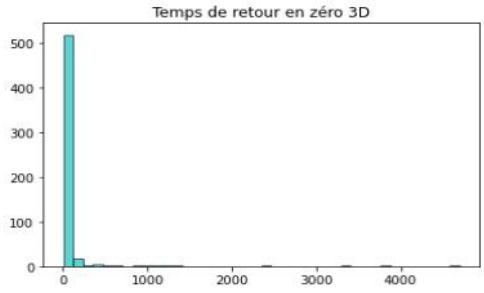

```
nombre de lancer : 10000
nombre de marche : 1000
il y a eu 2682 retour en zéro dans toutes les marches
**********
```





```
nombre de lancer : 10000
nombre de marche : 1000
il y a eu 561 retour en zéro dans toutes les marches
**********************
```





76047/561 = 136 Il y a ici 136 fois plus de retour en 0 pour la 1D que la 3D. On remarque, par ailleurs, que les retours en 0 sont effectués le plus souvent au début des marches.

Partie III : La Méthodes de travail

Utilisation de GIT :

Pour réaliser ce projet, nous avons dû nous mettre d'accord sur les tâches à réaliser et sur quels supports travailler pour pouvoir se partager le code de la manière la plus simple possible.

Pour la partie Python, nous avons créé un repository GIT. Ce choix a été très bénéfique car à chaque fois qu'un membre du groupe modifiait le code l'autre avait la possibilité de le récupérer très facilement.

Lien vers nôtre repository GitLab : https://gitlab.com/gaelbellanger283/Math-Proba

Utilisation de Google drive :

Concernant les résultats que l'on a pu obtenir, nous avons créé un dossier sur google drive avec tous les graphiques obtenus ainsi que les réponses aux questions du sujet.

Cela nous a permis de ne pas être perdus en début de séance et de savoir exactement là où nous nous étions arrêtés la fois précédente.

Utilisation de Discord :

Enfin, lorsque nous n'étions pas ensemble en séance de TP et que nous travaillions sur le projet, nous avons utilisé discord pour s'échanger des informations et se tenir au courant de l'évolution du projet.

Conclusion

Pour conclure, ce projet nous a permis de mieux comprendre le processus de marche aléatoire.

L'étude de son comportement en temps long met en valeur des éléments qui la caractérise.

En effet, il faut tout d'abord déterminer si la marche est « équilibrée » ou « non équilibrée ».

Si cette dernière est **équilibrée** il existe deux cas possibles :

- La dimension est strictement inférieure à 3 -> Marche Récursive
- La dimension est supérieure ou égale à 3 -> Marche transitive

C'est le principe du théorème de Polya, celui-ci étant un peu compliqué au premier abord nous avons effectué quelques recherches qui nous ont permis de mieux le comprendre…

En effet si l'on prend l'exemple de l'homme ivre pour la 2D et de l'oiseau pour la 3D alors le théorème de Polya permet de dire qu'un homme ivre se déplaçant dans un plan en 2 dimensions, parviendra toujours à rentrer chez lui alors qu'un oiseau ivre se déplaçant en 3 dimensions, pourrait se perdre pour toujours.

Dans le cas d'une marche **non équilibrée,** elle sera toujours transitive.

Ce projet nous a permis de découvrir Python que nous ne connaissions pas, de voir tout le potentiel de ce langage pour analyser des données notamment avec des graphiques. De plus il a amélioré nos compétences sur Git.