Rapport du projet de Simulation :

Schérer Robin Tondeur Pierre

Année académique : 2017-2018
Simulation
Université de Mons

19 mai 2018

Table des matières

1 Introduction

Ce projet à été réalisé dans le cadre du cours dispensé par Alain Buys, "Simulation sur ordinateur".

Le but de ce projet est de premièrement étudier le caractère pseudo-aléatoire des décimales de π (1.000.000 de décimales ont été étudiées) via des tests vue au cours. Nous devions ensuite utilisé ces décimales pour implémenter un générateur de loi uniforme $[0, 1[$ et ensuite le comparer avec le générateur par défaut de Python via les tests implémentés

2 Analyse des décimales de π

Tentons de déterminer si les décimales de π suivent bien une loi uniforme via les tests suivant.

2.1 Test du χ^2

Ce test statistique permet de tester l'adéquation d'une série de données à une famille de lois de probabilités. Pour ce test nous avons besoin d'un certain nombre d'intervalle, et nous comptons le nombre de valeurs qu'on a générées dans chaque intervalles, ceci est notre hypothèse. Nous avons aussi besoin une hypothèse nulle notée H_0 qui considèrent que les données suivent une lois de probabilité donnée et nous avons notre hypothèse qui sont nos données obtenu expérimentalement. Nous comparons ensuite ces 2 échantillons de données via la formule suivante :

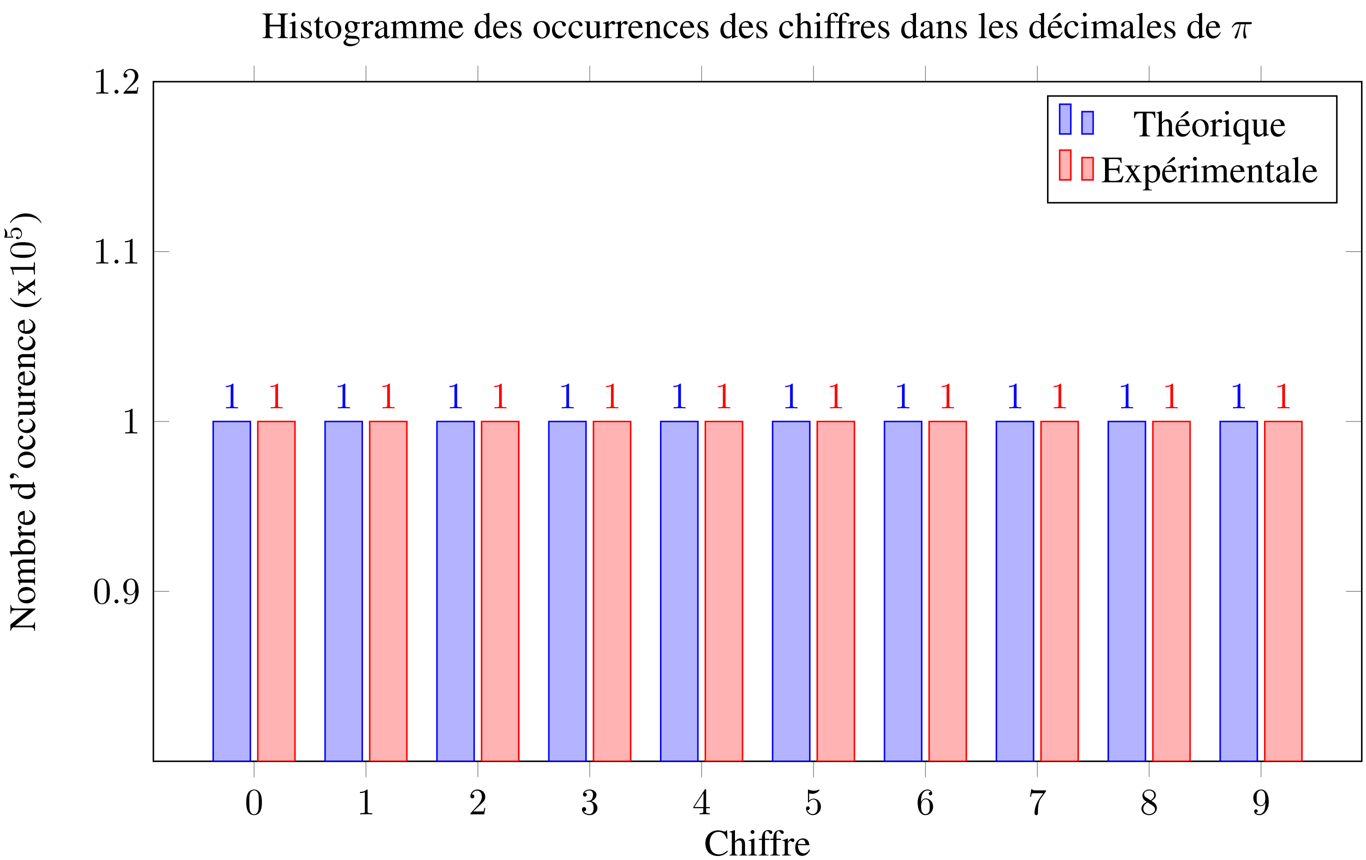
$$K_n = \sum_{i=1}^{r} (\frac{n_i - Np_i}{\sqrt{Np_i}})^2$$

on a que r est le nombre d'intervalle, Np_i est l'effectif théorique pour la classe i, et n_i est l'effectif qu'on à obtenu expérimentalement dans la classe i. Et nous avons un degrés de libertés égale à r - 1. Puis on se donne une probabilité α qui est le risque d'erreur, avec α et le degré de liberté, on obtient une valeur critique, C_i. Et si $K_n < C_i$, alors on rejette notre hypothèse.

Pour les décimales de π on choisit 10 intervalles chaque intervalle contient 1 chiffre. On a que pour H_0, les décimales de π suivent une loi uniforme, donc la probabilité que chacun des chiffres apparaissent le même nombre de fois.

Voici nos résultats pour notre hypothèse :

	Résulats		
Chiffre	Théorique	Expérimentaux	Erreur relative(%)
0	100 000	2.3	
1	100 000	2.3	
2	100 000	2.3	
3	100 000	2.3	
4	100 000	2.3	
5	100 000	2.3	
6	100 000	2.3	
7	100 000	2.3	
8	100 000	2.3	
9	100 000	2.3	



Maintenant effectuons le test du χ^2 sur nos 2 hypothèse :

α	K	C_i	On garde notre hypothèse
0,01	100 000	2.3	Oui
0.025	100 000	2.3	Oui
0.05	100 000	2.3	Oui

On a donc que les décimales de π passe bien le test du χ^2 pour nos alpha et que donc elles suivent bien une lois uniformes.

2.2 Test du Gap

Pour ce test on a une suite de nombre $u_1, u_2, ..., u_n$. On choisit $1 \leq a \leq b \leq 9etonmarqueceuxquitombentdans[a, b], quipourchaq$
$\frac{(b+1)-a}{10} deseproduire.Etons'intresseensuiteladistanceentre2nombresd'affilermarqus.Onsedonne$n$nombredeclasse, dansla$

L'hypothèse nulle est que pour chaque classe i on a : $p(1-p)^i$

On compare ensuite ces 2 hypothèses avec le test du χ^2 pour savoir si le test est réussi ou non.

Dans notre cas, on choisit de faire 10 fois ce test pour chacun des 10 chiffre, et que a chaque fois a = b. On a donc à chaque fois que p = 0.1.

Voici ce qu'on obtient pour le nombre d'occurrence de certaines longueurs de trou pour le chiffre 5.

Longueurs du trou	Nombre d'occurrences théorique	Nombre d'occurrences obtenu	Erreur relative (%)
1			
2			
3			
4			
5			
6 ... 21	...	...	...
22			
23			
24			
25 ... 24	...	...	...
57			
58			
59			
60			

Pour le test de χ^2 on a les résultats suivant :

α	K	C_i	On garde notre hypothèse
0,01	100 000	2.3	Oui
0.025	100 000	2.3	Oui
0.05	100 000	2.3	Oui

On a que pour les 10 chiffres tout les tests passe bien. On a donc que les décimales de π réussissent le test du Gap.

2.3 Conclusion

On a donc que les 1.000.000 de premières décimales de π suivent bien une loi uniforme.

3 Générateur de nombre pseudo-aléatoire avec les décimales de π

3.1 Principe de notre générateur

Pour notre générateur de nombre pseudo-aléatoire nous avons choisit d'utiliser la congruence linéaire, et le caractère pseudo-aléatoire sera le temps actuelle sur la machine. Nous avons donc une valeur pour a, c, et m utilisé pour la congruence linéaire, un index compris entre 0 et 1.000.000, et bien sûr les 1.000.000 de première décimales de π dans un string.

Les valeurs choisies pour la congruence linéaire sont a = 41, c = 11, et m = 1.000.000. Ces valeurs respectent le Théorème de Hull et Dobel pour maximiser la période pour revenir à notre index de départ. On a que m = 1.000.000 car les index pour π vont de 0 à 999.999, on doit aussi avoir que c est premier avec m, donc 11 est premier avec m. On doit aussi avoir que b est un multiple de 4 si 4 est un diviseur de m, ce qui est le cas. On a que a = b + 1, et que b est un multiple de p $\forall$ p diviseur premier de m. On a donc que 2 et 5 sont les diviseurs premier de m, et donc b est un multiple de 2, 4 et 5. On a choisit b = 40, et donc a = 41.

Quand on initialise le générateur, il initialise l'index de base avec le temps actuelle avec la méthode time.time(). À chaque fois qu'on demandera un nombre à notre générateur, il assignera $(((a * index) + c)\%m)$ à l'index et retournera les n (16 par défaut) chiffres à partir de cet index dans les décimales de π.