Simulation sur ordinateur
Première session : Rapport
Etude du caractère pseudo-aléatoire de π

Maazouz Mehdi, Caredda Giuliano
BA3 Info

28 Mai 2018

Sommaire

1 Introduction

Le travail que nous devons présenter consiste à étudier le caractère pseudo-aléatoire des décimales de π en utilisant les techniques vues au cours et ce, suivant une lois uniforme.
De plus, nous allons devoir nous servir de ces décimales pour implémenter un générateur de loi uniforme [0,1[et de comparer ce dernier au générateur par défaut de Python.
Nous effectuerons 3 tests pour l'étude du caractère pseudo aléatoire et 3 autres tests pour la comparaison avec le générateur par défaut de Python. Ces derniers seront décris ci-dessous.
Nous nous servirons du langage Python pour effectuer nos différents tests.

1.1 Fichiers Annexes

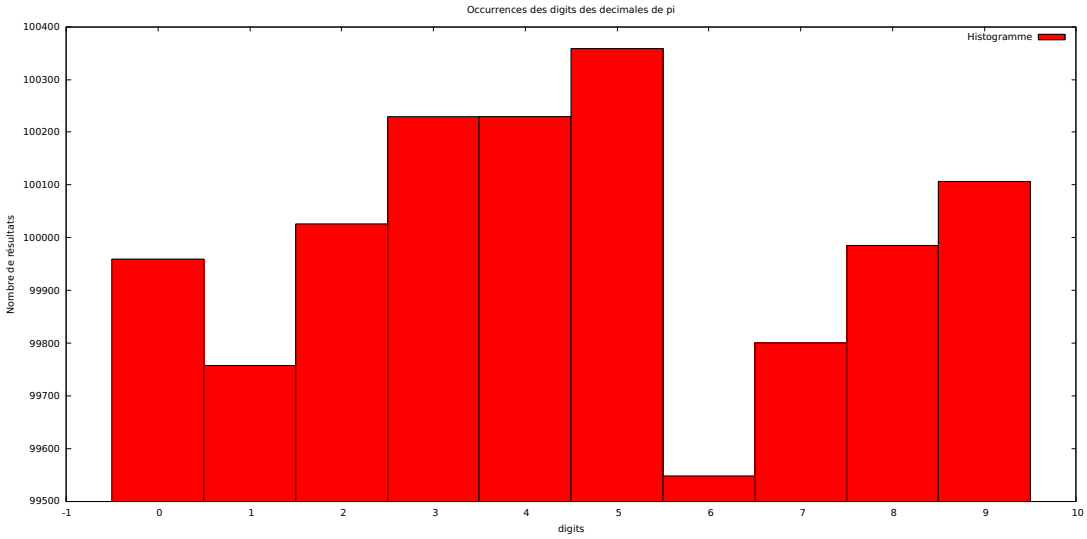
Afin de réaliser ce projet, nous allons devoir réaliser quelques scripts pour les différents tests qui vont suivre ci-dessous. Pour ce faire, nous nous sommes servis du langage de programmation Python dans sa version 2.7.
Son module Gnuplot va également servir pour les graphes qui seront tracés au fur et à mesure. D'autres modules de Python tel que le module Math ou Sympy ont été utilisés.

2 Etude de π

2.1 Approche Naïve

Nous savons que π contient un nombre infinis de décimales. En partant de ça, supposons que nos décimales suivent une loi uniforme. En partant de cette supposition, nous allons tracer un histogramme comprenant les occurrences de chaque digit parmi le million de décimales fournies sur la plateforme Moodle.
Le nombre d'occurrence théorique de chaque digit devrait être de 100 000. En effet, on a 10 digits (de 0 à 9), avec une probabilité de 1/10 de sortir.

Regardons maintenant notre premier histogramme et débattons ensuite des premières constatations.



Mettons désormais les données sous la forme d'un tableau.

	Résultats		
Digit	**Observé**	**Attendu**	**Erreur en %**
0	99 959	100 000	0.041
1	99 758	100 000	0.24
2	100 026	100 000	0.026
3	100 229	100 000	0.229
4	100 230	100 000	0.230
5	100 359	100 000	0.359
6	99 548	100 000	0.452
7	99 800	100 000	0.2
8	99 985	100 000	0.015
9	100 106	100 000	0.106

On peut constater qu'entre les résultats observés et attendus, il y a à chaque fois une erreur inférieur à 0,5%
Ce qui nous encourage à penser que nos données suivent bien une loi uniforme même si cela ne constitue pas une preuve.
Nous allons bientôt passer à l'étude du caractère pseudo-aléatoire des décimales de π par l'intermédiaire de 3 tests différents.

2.2 Test de χ^2

Le premier test que nous allons effectuer est le test du χ^2. Ce dernier est un test statistique qui va permettre de vérifier si une série de données est susceptible

de suivre une loi de probabilité.
Pour effectuer ce test, nous allons devoir regrouper ces données dans différentes classes. Voici un rappel théorique :

$$K_n = \sum_{i=1}^{r} (\frac{n_i - Np_i}{\sqrt{Np_i}})^2 \; et \; on \; prend \; \chi^2_{r-1,1-\alpha}$$

On peut y apercevoir r qui correspond au nombre de classes.
Np_i qui correspond aux nombres d'occurrences théoriques.
n_i qui correspond aux nombres d'occurrences observées.

Partons désormais de notre hypothèse nulle nommée H_0 = nos décimales de π suivent une loi uniforme.
Dans notre cas, r sera fixé à 10, car nous considérerons une classe par digit, et Np_i sera fixé à 100 000.
Quant à n_i, ses différentes valeurs seront issues du nombre d'occurrence observés dans l'histogramme présenté précédemment. Comme le demande le test de χ^2, nous devons fixer une probabilité α, appeler erreur de première espèce, de rejeter l'hypothèse nulle H_0.
Une fois ce paramètre α fixé, nous pouvons désormais vérifier si H_0 est acceptée si $K_n <= \chi^2_{9,\alpha-1}$ où 9 est le degré de liberté calculé en faisant r-1 (nombre de classes -1).

Voici les différents résultats obtenus en fonction de différentes valeurs pour α.

	Résultats		
α	K_n	$\chi^2_{9,\alpha-1}$	**Resultat**
0.001	5.50908	27.877	Reussite
0.025	5.50908	19.023	Reussite
0.1	5.50908	14.684	Reussite

On peut constater que les tests réussis, nous sommes donc confortés dans notre supposition qui était que les décimales de π suivent une loi uniforme.

2.3 Test du Poker

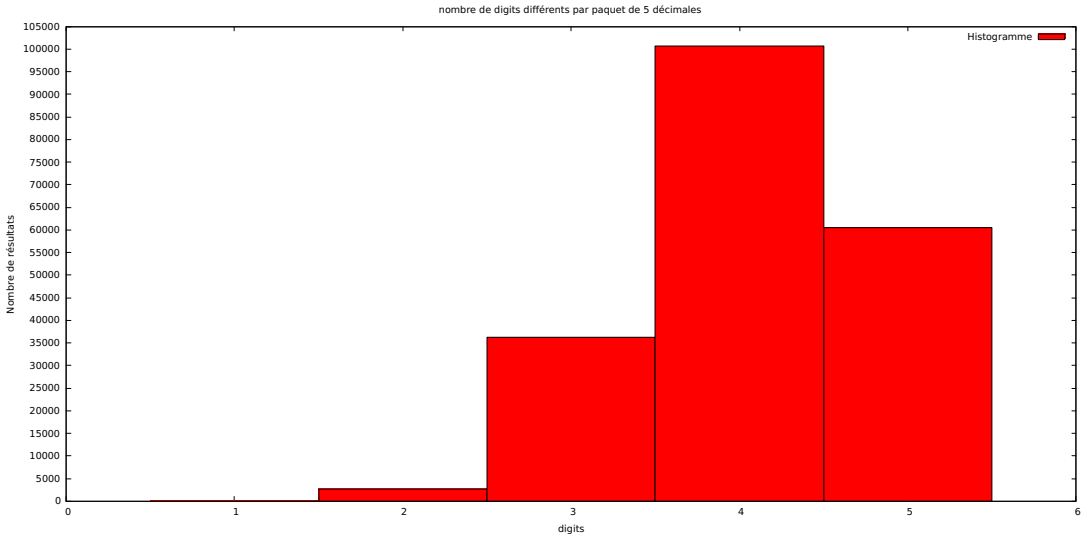
Une fois le test du χ^2 effectué, passons maintenant à un autre test qui va s'appuyer sur ce dernier, le test du Poker. Pour ce faire, nous allons diviser notre million de décimales en 200 000 blocs contenant chacun 5 décimales. Nous dirons que chaque bloc contient une séquence.

Une fois les blocs obtenus, nous allons les classer en fonction du *nombre* de digit différents par bloc. Ce *nombre* est appelé r et est compris entre 1 et 5.
En effet, une séquence peut seulement contenir un digit différent (par exemple : 11111) mais également 5 digits différents (12345).
Nous allons désormais passer au calcul du χ^2 en se basant sur le nombre de *Stirling* pour effectuer la comparaison. Ce dernier se définit comme étant le *nombre de manières possibles de constituer r paquets avec k nombres.*
Par le rappel théorique (*voir cours*), nous obtenons P_r, la probabilité d'avoir r.

$$P_r = \frac{\left\{ \begin{array}{c} k \\ r \end{array} \right\} d(d-1)...(d-r+1)}{d^k} \qquad (r \leq d)$$

Avec **K** la longueur de la séquence , **d** le nombre de digits possibles et enfin **r** définis plus haut.
Pour effectuer nos calculs, nous fixerons **K** à 5 et **d** à 10.
Une fois les probabilités obtenues, multiplions ces dernières par le nombre de paquet (200 000).

r	valeur théorique	valeur calculée
1	20	13
2	2700	2644
3	36000	36172
4	100800	100670
5	60480	60501



3 Générateur de nombre pseudo-aléatoire

3.1 Enonce

La deuxième partie de notre projet consiste à implémenter un générateur de nombre pseudo-aléatoire et ensuite, de le comparer avec le générateur par défaut de Python (Mersenne Twister). Plus concrètement, nous devons implémenter un générateur qui devra, comme son nom l'indique, générer des nombres compris entre 0 et 1 de manière uniforme.

3.2 Implémentation

Nous savons que l'ordinateur est une machine déterministe. Cependant, nous pouvons faire ressortir une composante aléatoire en utilisant le temps écoulé, en millisecondes, depuis le 1^{er} Janvier 1970.
Etant donné que les premières décimales de π nous sont fournies, nous allons pouvoir nous en servir pour l'implémentation de notre générateur.