Probabilités pour l'Informatique Enseignant : Brice OLIVIER
TP 1 - générateurs de nombres pseudo-aléatoires
Ensimag 2A - Apprentissage
1er semestre 2016 - 29/09/2016
Deadline : Mercredi 5/09/2016 - 23h59

Libraires requises

— python 2.7
— matplotlib

Quelques outils

— ipython - terminal python enrichi
— jupyter notebook - document alliant ipython et texte enrichi pour des présentations un peu fancy
— spyder - IDE pour la programmation scientifique en python. Equivalent de Rstudio pour R
— vi(m) ou emacs - pour les barbus (sans offense)
— PEP 8 - documentation pour écrire du code python standard

Remarque : Libre à vous de programmer en orienté objet ou non

---

Exercice 1 - Algorithme de John Von Neumann (1946) - Middle-square method

1. Choisir $x_0 \in \mathbb{N}^* \cap x_0 \leq N$, où N est le nombre maximum composé de n chiffres, $\forall n \mod 2 = 0$
2. Calculer $y_n = x_n^2$. Puis, compléter y_n par autant de 0 à gauche pour obtenir un nombre à n chiffres
3. Affecter à x_n le "milieu du carré" de y_n. C'est à dire, supprimer les $n/4$ chiffres à droites et les $n/4$ chiffres à gauche de y_n
4. Retourner à l'étape 2 pour un nombre donné n_iter d'itérations

Exemple : $n = 8, n\_iter = 2$
1. $x_0 = 2964$
2.1. $y_1 = 2964^2 = 08785296$
3.1. $x_1 = 7852$
2.2. $y_2 = 7852^2 = 61653904$
3.2. $x_2 = 6539$

Q1) Programmer l'algorithme.

Q2) Testez l'algorithme avec quelques valeurs différentes. Que remarquez-vous ?

Q3) Vérifiez les propriétés d'un bon générateur de nombres pseudo-aléatoires par des méthodes graphiques de statistique descriptive. Qu'en pensez-vous ?

Un an plus tard, au cours d'une conférence, Von Neumann déclare : "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."

Exercice 2 - Algorithme de Green, Smith, Klem (1959) - Additive Lagged Fibonacci Generator

1. Choisir un graine $\{x_0, ..., x_j, ..., x_k, x_{n-1}\}, 0 < j < k, \forall n, x_n \in \mathbb{N}$
2. $x_n = (x_{n-j} + x_{n-k}) \mod m, m \in \mathbb{N}^*$

Exemple : $m = 11, j = 1, k = 3$
1. $x_0 = 3, x_1 = 4, x_2 = 7, x_3 = 9, x_4 = 10$
2.1. $x_5 = (x_{5-1} + x_{5-3}) \mod 11 = 6$
2.2. $x_6 = 4$
2.3. $x_7 = 3$
2.4. $x_8 = 9$
2.5. $x_9 = 2$

Q1) Programmer l'algorithme.

Q2) Testez l'algorithme avec quelques valeurs différentes.

Q3) Vérifiez les propriétés d'un bon générateur de nombres pseudo-aléatoires par des méthodes graphiques de statistique descriptive. Qu'en pensez-vous ?

Exercice 3 -Algorithme de Derrick Lehmer (1948) - Congruential Linear Generator

Q1) Programmer le CLG vu en cours.

Q2) Vérifier vos calculs papiers effectués lors du précédent cours.

Q3) Pour $m = 2048, x_0 = 0$ et les valeurs suivantes de a et c, faites un scatter plot de 512, 1024 et 2048 valeurs successives des paires (x_n, x_{n+1})

1. $a = 65, c = 1$
2. $a = 1365, c = 1$
3. $a = 1229, c = 1$
4. $a = 151, c = 1$
5. $a = 45, c = 0$, prendre $x_0 = 1$
6. $a = 43, c = 0$, prendre $x_0 = 1$

Q4) Quel(s) cas vous semble(nt) conforme(s) à ce qu'on l'on devrait attendre d'une variable aléatoire uniforme indépendante ?

Q5) Pour les cas (a) et (e), représenter en plot3D les valeurs successives de $(x_n; x_{n+1}; x_{n+2})$. Quelle conclusion en tirez vous ?

Exercice 4 - Générateurs combinés

Q1) Programmer la combinaison de générateurs congruentiels linéaires (slide 17)

Q2) Programmer la générateur à récursivité multiple (slide 18)

Q3) Testez les générateurs pour quelques valeurs

Q4) Faites un scatter plot des données $(x_n; x_{n+1})$. Que constatez-vous ?

Exercice Bonus

Dans le cadre de ce premier chapitre, nous avons étudié les générateurs de nombres pseudo-aléatoires. Il existe une autre classe de générateurs aléatoires appelée Hardware/True Random Number Generator. Ces générateurs se basent sur l'observation de phénomènes physiques dits "aléatoires".

Q1) En quelques lignes (20 lignes grand maximum), dites moi ce que vous voulez sur le sujet en mettant en opposition ces deux familles de générateurs