

Filières MP/MPI - ENS de Paris-Saclay, Lyon, Rennes et Paris - Session 2024

Page de garde du rapport de TIPE

NOM : Montfort	Prénoms : Perig Louis
Classe : MPI	
Lycée : Faidherbe	Numéro de candidat : 11965
Ville : Lille	

Concours auxquels vous êtes admissible, dans la banque MP inter-ENS (les indiquer par une croix) :

ENS Cachan	MP - Option MP		MP - Option MPI	
	Informatique		Informatique	X
ENS Lyon	MP - Option MP		MP - Option MPI	
	Informatique - Option M		Informatique - Option P	X
ENS Rennes	MP - Option MP		MP - Option MPI	
	Informatique		Informatique	X
ENS Paris	MP - Option MP		MP - Option MPI	
	Informatique			

Matière dominante du TIPE (la sélectionner d'une croix inscrite dans la case correspondante) :

Informatique	X	Mathématiques		Physique	
--------------	---	---------------	--	----------	--

Titre du TIPE : Générations et tests de suites pseudo-aléatoire pour la création d'un jeu de poker.


Nombre de pages (à indiquer dans les cases ci-dessous) :

Texte	6	Illustration	1	Bibliographie	1
-------	---	--------------	---	---------------	---

Attention, les illustrations doivent figurer dans le corps du texte et non en fin du document !

Résumé ou descriptif succinct du TIPE (6 lignes, maximum) :

Dans le cadre de la création d'un jeu de poker, la génération de l'aléatoire dans la distribution des cartes a été mon principale objectif. Comment générer de l'aléatoire à partir d'un algorithme alors que celui-ci est déterministe ? Nous étudierons les suites pseudo-aléatoires en les générant et en concevant une batterie de tests permettant de déterminer celles qui sont les plus adaptées à notre jeu.

À Lille	Signature du professeur responsable de la classe préparatoire dans la discipline	Cachet de l'établissement
Le 05/06/2024		
Signature du (de la) candidat(e)		

Rapport de TIPE ENS

Générations et tests de suites pseudo-aléatoires

Création d'un jeu de poker

Introduction

L'aléatoire et sa génération sur ordinateur est un problème très important actuellement, tant dans son utilité dans le cryptage de données en cryptographie que dans la simulation de phénomènes physiques comme la météorologie. Dans notre situation, celui-ci est important pour le mélange des cartes d'un jeu de poker, qui doit permettre de tirer uniformément des cartes de manière imprévisible. La difficulté de mettre en place une génération par des méthodes physique nous pousse à trouver une méthode algorithmique plus efficace. Le problème est qu'on ne peut de manière algorithmique que *simuler* l'aléatoire à cause du déterminisme de l'algorithme. Dès lors, il existe différentes suites dites pseudo-aléatoires qui permettent de simuler les propriétés de l'aléatoire et d'une distribution uniforme mais l'on peut se demander dans quelle mesure celles-ci peuvent-elles être considérées comme aléatoires.

Dans cet exposé, nous nous intéresserons à trois suites pseudo-aléatoires : *middle square* par Von Neumann, *Linear Congruential Generator* par Lehmer ainsi que *Blum Blum Shub* du même nom que ses auteurs. Nous analyserons les propriétés de ces suites pour finalement réaliser différents tests qui détermineront l'efficacité de celles-ci dans diverses situations.

0. Mélange par la méthode de l'algorithme P

Pour mélanger les cartes par la suite, on utilisera le mélange de Knuth.

L'algorithme permet de mélanger un tableau de taille n de manière uniforme en complexité linéaire par rapport à n . Donc ici en temps constant puisque n sera fixé égal à 52. L'algorithme est le suivant :

Pour i allant de 0 à $n-1$:

choisir un j au hasard entre 0 et i
échanger $t[i]$ et $t[j]$

Il ne nous manque plus qu'à trouver une méthode pour choisir ce j "au hasard".

1. Middle Square

Le premier générateur est celui de la méthode *middle square* et son principe est tel que :

Pour générer $(x_n)_n$ on pose un x_0 arbitraire et pour tout $n \in \mathbb{N}$, on obtient le terme x_{n+1} en prenant les chiffres du milieu de x_n^2 .

On s'intéresse à des nombres composés de k chiffres dont le carré fera $2k$ chiffres (quitte à ajouter des zéros). Par exemple pour $k = 4$ en commençant par $x_0 = 2345$ on a $x_0^2 = 05499025$ donc $x_1 = 4990$ puis en répétant le procédé on a 9001, 180, 3240, 4976, 7605, 8360, 8896, 1388, 9265...

On remarque que son fonctionnement fait que la périodicité de la suite est très faible par rapport à 10^k . En effet, si un nombre apparaît deux fois on entre dans un cycle. On peut approximer la probabilité d'avoir un cycle avant le j -ième terme en supposant que x_n est un nombre tiré uniformément dans $\llbracket 0 ; 10^k - 1 \rrbracket$.

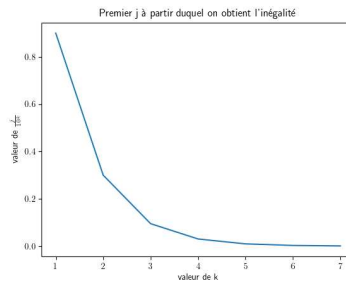
On note X_j : "Le nombre de fois qu'on a tiré un doublon en j tirages"

L'événement qui nous intéresse est donc $[X_j > 0]$ mais on va calculer son complémentaire $[X_j = 0]$.

Il y a en tout 10^{kj} tirages possibles dont $j!(10^k)$ sans doublons. Dès lors $P(X_j > 0) = 1 - \frac{10^k!}{(10^k-j)!10^{kj}}$

Alors on peut obtenir le plus petit j tel que $P(X_j > 0) \geq 0.99$ donc que $0.01 \geq \frac{10^k!}{(10^k-j)!10^{kj}}$

On peut réécrire le membre de droite en : $\prod_{i=0}^{j-1} (1 - \frac{i}{10^k})$ ce qui va simplifier la résolution algorithmique.



On remarque donc que plus on augmente k , plus j devient petit par rapport à 10^k et moins le générateur sera efficace.

2. Générateur à Congruence Linéaire

Le second générateur est le *Générateur à Congruence Linéaire* (GCL) dont le principe est le suivant :

Soit $m \in \mathbb{N}$, soit $(a, c, X_0) \in \llbracket 0, m-1 \rrbracket^3$ où m est le module, a le multiplicateur, c l'incrément et X_0 la graine, on génère (x_n) par $x_0 = X_0$ et $\forall n \in \mathbb{N}, x_{n+1} = (ax_n + c)[m]$

Pour étudier le GCL. On commence par une formule explicite des termes permettant de ne pas toujours commencer la partie au terme X_0 :

Théorème 1 Soient $n \in \mathbb{N}$ et $k \in \mathbb{N}^*$. Alors : $X_{n+k} = (a^k X_n + c \frac{(a^k - 1)}{a - 1})[m]$

Ce qui se montre par récurrence immédiate sur k .

Par la suite, on veut choisir efficacement a , c , x_0 et m pour obtenir un générateur efficace.

En effet, par exemple pour $a = 4$, $c = 1$, $m = 64$ et $x_0 = 5$:

On a $x_1 = (4 \times 5 + 1)[64] = 21$, $x_2 = (21 \times 4 + 1)[64] = 85[64] = 21$, et donc $x_3 = 21$ et ainsi de suite.

La congruence modulo 2 étant la plus rapide sur ordinateur, elle paraît être le premier choix pour m . De plus un théorème sur la périodicité existe pour $m = 2^n$

Théorème 2 Soit λ la périodicité : $a \equiv 1[4] \wedge c \equiv 1[2] \Leftrightarrow \lambda = 2^n$

Pour démontrer ce théorème on a besoin d'un lemme intermédiaire disponible en annexe.

Lemme 1 Soient $p \in \mathbb{P}, k \in \mathbb{N}, p^k > 2$: $x \equiv 1[p^k] \wedge x \not\equiv 1[p^{k+1}] \implies x^p \equiv 1[p^{k+1}] \wedge x^p \not\equiv 1[p^{k+2}]$

On commence par le sens direct :

On suppose $X_0 = 0$ sans pertes de généralités, $a \equiv 1[4]$ et $c \equiv 1[2]$.

$\exists q \equiv 1[2], \exists k \geq 2, a = 1 + 2^k q$. Dès lors $a \equiv 1[2^k]$ et $a \not\equiv 1[2^{k+1}]$

En appliquant le lemme :

$\forall k' \in \mathbb{N}, a^{2^{k'}} \equiv 1[2^{k+k'}]$ et $a^{2^{k'}} \not\equiv 1[2^{k+k'+1}]$

Donc on a d'après le théorème de Gauss : $\frac{a^{2^{k'}}-1}{a-1} \equiv 0[2^{k'}]$

En particulier pour $k' = n$: $X_{2^n} \equiv c \frac{(a^{2^n}-1)}{a-1} \equiv 0[2^n]$

Ce qui signifie qu'il existe $k' \leq n$ tel que $\lambda = 2^{k'}$

Si $k' < n$: $\frac{(a^{2^{k'}}-1)}{a-1} \not\equiv 0[2^{k'+1}]$

Donc : $\frac{(a^{2^{k'}}-1)}{a-1} \not\equiv 0[2^n]$ et $c \not\equiv 0[2] \implies X_\lambda \not\equiv 0[2^n] \implies$ Absurdité

Dès lors : $\lambda = 2^n$

Pour le sens réciproque :

On suppose $X_0 = 0$ sans pertes de généralités et $\lambda = 2^n$

Si $c \equiv 0[2]$: $\forall k, X_k \equiv c \frac{(a^k-1)}{a-1} \equiv [m]$ donc X_k est pair.

Il y a donc au plus 2^{n-1} valeurs de X_k et par principe des tiroirs il y en a deux égaux \implies Absurde

On suppose donc c impair.

Si $a \equiv 0[2]$:

$\frac{a^{2^n}-1}{a-1} \equiv 0[2^n] \implies a^{2^n} - 1 \equiv 0[2^n] \implies a^{2^n} \equiv 1[2^n] \implies$ Absurde

Si $a \equiv 3[4]$:

Montrons par récurrence h_k : $\forall k > 1, a^{2^{k-1}} \equiv 1[2^{k+1}]$

Initialisation : Hypothèse : $\exists j \in \mathbb{N}, a = 3 + 4j$

$a^2 = 9 + 24j + 16j^2 \equiv 1[8]$

Hérédité : Soit $k > 1$, on suppose H_k vraie. Par hypothèse de récurrence : $\exists q \in \mathbb{N}, a^{2^{k-2}} = 1 + 2^k q$
 $a^{2^{k-1}} = a^{2^{k-2} \times 2} = (a^{2^{k-2}})^2 = (a^{2^{k-2}} + 1)(a^{2^{k-2}} - 1) + 1$

Donc $a^{2^{k-1}} = (2 + 2^k q)(2^k q) + 1 = (1 + 2^{k-1} q)(2^{k+1} q) + 1 \implies a^{2^{k-1}} \equiv 1[2^{k+1}]$. H_{k+1} reste vraie.

Dès lors pour $k = n$, par principe de récurrence, $a^{2^{n-1}} - 1 \equiv 0[2^{n+1}] \implies X_{n-1} = \frac{a^{2^{n-1}}-1}{a-1} \equiv 0[2^n]$

Absurde donc $a \equiv 1[4]$ et $c \equiv 1[2]$.

Ce théorème se généralise pour un m quelconque :

Théorème 3 Soit λ la périodicité, $m = m_1^{\alpha_1} \dots m_d^{\alpha_d}$ la décomposition en produit de facteurs premiers de m , alors $\lambda = m$ si et seulement si :

- $c \wedge m = 1$
- $a - 1$ est multiple de m_i , $\forall i \in \llbracket 1 ; d \rrbracket$
- si m est multiple de 4, $a - 1$ est multiple de 4

Ce qui peut être utile par la suite si on rejette les générateurs où $m = 2^n$.

3. Blum Blum Shub

Le dernier générateur étudié est le générateur *Blum Blum Shub* qui nécessite d'abord d'introduire une suite définie par p et q deux nombres premiers grands congrus à 3 modulo 4, $N = pq$, $x_0 \in \llbracket 0 ; N - 1 \rrbracket$ tel que $x_0 \wedge N = 1$ et $\forall n \in \mathbb{N}, x_{n+1} = x_n^2[N]$

On peut ensuite créer la suite de Blum Blum Shub (u_n) tel que pour tout n , u_n soit le bit faible de x_n

(c'est-à-dire la congruence modulo 2).

Comme le GCL, il est aussi possible de commencer à un terme x_k la suite puisque par récurrence immédiate, $x_k = x_0^{2^k} [N]$. Cependant le coût de l'opération sera assez lourd puisque l'algorithme d'exponentiation rapide pour un exposant 2^k est en $O(k)$.

Mais en prenant la division Euclidienne de 2^k par $\phi(N)$ où ϕ est l'indicatrice d'Euler, on a $x_0^{2^k} = x_0^{2^k[\phi(N)]} x_0^{\rho\phi(N)}$. Et puisque d'après le théorème d'Euler valide par l'hypothèse de primalité entre x_0 et N , $x_0^{\phi(N)} \equiv 1[N]$ on obtient le théorème :

Théorème 4 $\forall k \in \mathbb{N}, x_k = x_0^{2^k[(p-1)(q-1)]} [N]$

Permettant d'obtenir ce terme en $O(\log(k))$ puisque la puissance de x est bornée.

On peut encore réduire le modulo grâce à la fonction de Carmichael λ donnant le plus petit entier m tel que pour tout x , $x^m \equiv 1[m]$ qui vaut $\lambda(N) = \frac{(p-1)(q-1)}{(p-1) \wedge (q-1)} = (p-1) \vee (q-1)$ dans notre cas.

En considérant le groupe engendré par 2 sur $\mathbb{Z}/\lambda(N)\mathbb{Z}$ on remarque d'après le théorème de Lagrange que la période du générateur divise $\lambda(N)$ (on cycle lorsque $x_k \equiv x_0[N]$ donc que $2^k \equiv 1[\lambda(N)]$). Il faut donc minimiser $\phi(p-1) \wedge \phi(q-1)$ pour maximiser la période du générateur.

L'avantage de ce générateur est que si l'on ne connaît pas la décomposition de N , on ne peut pas retrouver le terme précédent le rendant sûr cryptographiquement.

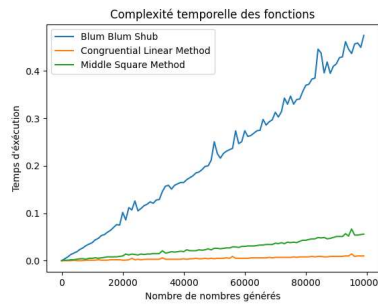
En effet, si l'on peut retrouver le terme précédent en temps polynomial, et qu'on trouve un x tel que y soit l'appel de $BBS(x^2)$ et que $y \not\equiv \pm x[N]$, alors N ne divise ni $(x-y)$ ni $(x+y)$.

Or N divise $(y^2 - x^2)$ donc N divise $(x-y)(x+y)$ donc $x \pm y \wedge n$ est un diviseur de N non trivial.

Dès lors on a résolu le problème de la factorisation de N en temps polynomial ce qui est absurde en considérant que ce problème est conjecturé comme n'appartenant pas à P.

4. Test des générateurs.

Tout d'abord un test sur la complexité des fonctions nous donne ce graphique :



Ce qui correspond aux valeurs attendues puisque Blum Blum Shub doit générer bit par bit et Middle Square doit décomposer les chiffres de x_n pour y extraire le nombre au milieu.

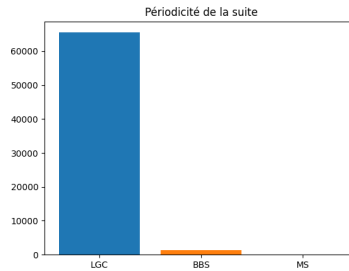
Un deuxième test est réalisé pour obtenir la périodicité de la suite. Pour cela on va appliquer l'algorithme de Pollard plus communément appelé algorithme du lièvre et de la tortue.

Son principe repose sur le fait qu'une valeur de notre suite x_{n+1} dépend uniquement de x_n donc si on trouve $x_i = x_j$ avec $i \neq j$ alors une boucle est créée.

Pour se faire l'algorithme va placer un indice i étant considéré comme la tortue et un indice j comme étant le lièvre. au début de l'algorithme $i = 1, j = 2$ et à chaque tour de boucle on augmente i de 1 et j de 2. si

$x_i = x_j$ alors on renvoie j-i qui correspond bien à la longueur du cycle obtenu.

Pour optimiser la complexité en espace et ne pas créer trop de valeurs on pourra utiliser nos bornes trouvées précédemment et on obtient alors généralement un graphique de la forme :



On remarque que la différence de périodicité est très marquée par rapports aux suites. On retrouve aussi que la suite Middle Square à une périodicité très faible et que le GCL à une périodicité égale à m .

Ensuite, nous allons définir un test d'adéquation qui s'appelle le test du chi-carré.

Théorème 5 Soit $n \in \mathbb{N}$, soit $p = (p_i)_{i \in \llbracket 1 ; n \rrbracket}$ et $q = (q_i)_{i \in \llbracket 1 ; n \rrbracket}$ deux vecteurs. On définit l'adéquation de p par rapport à q , χ^2 , par :

$$\chi^2(p, q) = \sum_{i=1}^n \frac{(p_i - q_i)^2}{q_i}$$

p désignera les valeurs empiriques et q les valeurs théoriques obtenues par une loi X .

On note D le degré de liberté de χ^2 et on pose un risque α d'erreur.

On peut comparer nos valeurs obtenues par rapport au tableau du chi carré disponible en annexe.

Nous allons d'abord appliquer ce test à une répartition uniforme. Pour cela, on va pouvoir réaliser un test Spectral :

Théorème 6 Le test Spectral sur une suite $(u_n)_{n \in \mathbb{N}}$ en dimension k consiste à étudier la répartition du nuage de points $(u_n, u_{n+1}, \dots, u_{n+k-1})$ pour tout n

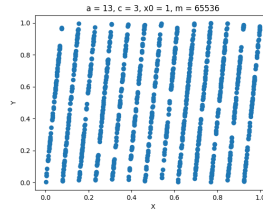
Par exemple pour $k = 1$ cela consiste à regarder la répartition des valeurs u_n dans l'intervalle $\llbracket 0 ; m - 1 \rrbracket$. En réalisant ce test en prenant les 1000 premiers nombres du générateur congruentiel linéaire $x_0 = 1, a = 67, c = 1, m = 2^{12}$ où on pose $h = \frac{m}{8}$ on a :

Intervalle	Valeurs dedans	Nombre attendu	Ecart
$[0; h - 1]$	104	125	3.528
$[h; 2h - 1]$	134	125	0.648
$[2h; 3h - 1]$	116	125	0.648
$[3h; 4h - 1]$	131	125	0.288
$[4h; 5h - 1]$	143	125	2.592
$[5h; 6h - 1]$	120	125	0.200
$[6h; 7h - 1]$	134	125	0.648
$[7h; m - 1]$	118	125	0.392
Total :	1000	1000	8.944

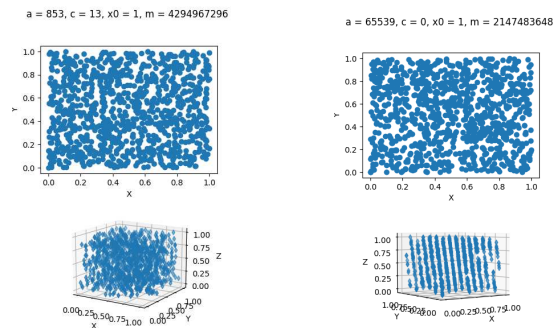
Ici $8.944 \leq 14.067$ donc le test est validé.

Ce test peut très bien se réaliser aux dimensions supérieures et même se visualiser pour les dimensions 2 et 3.

Par exemple voici un test que j'ai réalisé et qui a été non concluant en dimension 2 :



On peut aussi obtenir des tests valides en dimension 2 mais pas en 3 comme pour le graphique obtenu à droite:



Celui de gauche lui représente un test bien validé dans les deux dimensions.

Ce test a permis de détecter beaucoup de suites ayant des biais pour de faibles valeurs de k .

La totalité des suites générées par la méthode de Von Neumann ont été éliminées, environ 5% des GCL et moins de 1% pour Blum Blum Shub. Poursuivre le test pour de grandes valeurs de k était trop coûteux en temps puisqu'il fallait générer plus de valeurs et donc augmenter m et on peut ainsi en venir au dernier test.

Pour finir, nous allons réaliser un test sur la distribution des cartes au jeu de poker qui cette fois-ci n'est pas une distribution uniforme. Un exercice de dénombrement nous donne le tableau disponible en annexe.

On choisit de faire ce test sur 2 millions de parties à cause du critère de Cochran selon lequel il faut que toutes les valeurs théoriques soient supérieures à 1 et que 80% d'entre elles doivent être supérieures à 5.

Voici par exemple le tableau obtenu pour le GCL $a = 121, c = 1, m = 2^{32}$:

Main	Empiriques	Théoriques	Ecart
Quinte flush royale	2	3.08	0.378
Quinte flush	70	27.7	64.058
Carré	556	480	12.033
Full	2 755	2 881	98.958
Couleur	8 353	3 930	4942.784
Quinte	7 059	7 849	77.801
Brelan	42 368	42 256	1.998
Deux paires	95 078	95 080	3.477
Paire	843 018	845 138	5.332
Carte haute	1 000 164	1 002 354	4.811
Total	2 000 000	2 000 000	5211.629

On remarque que pour les couleurs, ce générateur n'a pas été concluant ! Alors que celui-ci passait bien le test précédent.

On peut expliquer cela par le fait que si on prend $d < 32$, et qu'on crée la suite (y_n) telle que $y_0 = x_0[2^d]$ et $\forall n \in \mathbb{N}, y_{n+1} = (ay_n + c)[2^d]$ on a pour tout n , $y_n \equiv x_n[2^d]$.

En effet l'initialisation est donnée par la définition et si la propriété est vraie au rang $n \in \mathbb{N}$ alors :

$$x_{n+1} = ax_n + c + qm \equiv (ax_n + c)[2^d]$$

Donc par hypothèse de récurrence : $x_{n+1} \equiv (ay_n + c)[2^d]$

Donc on a bien l'hérédité : $y_{n+1} \equiv x_{n+1}[2^d]$

Ce qui signifie que les d premiers bits significatifs des éléments de la suite ont un cycle de 2^d . Donc que la congruence modulo 4 à un cycle de 4 seulement. Puisque la couleur était décrite par la congruence modulo 4 on retrouve donc de manière plus apparente celle-ci.

Il faut donc prioriser un choix de m de la forme $m = 2^n \pm 1$ avec m ayant des facteurs premiers assez grands pour ne pas détecter facilement ce biais.

Pour ce qui est de ces générateurs et de ceux de Blum Blum Shub, on observe finalement des résultats satisfaisant le test du chi carré.

Conclusion

Pour conclure, les suites qui semblent les plus efficaces pour mon jeu de poker sont celles de Blum Blum Shub où celles du GCL avec $m = 2^n \pm 1$ et a et c qui suivent le théorème sur la périodicité.

D'un côté celles de Blum Blum Shub semblent plus sûres dans leur génération et sont très efficaces puisqu'elles n'ont pas échouées aux tests, de l'autre, contrairement à celles du GCL, elles sont très lentes à être générées et N doit être très grand pour obtenir une période suffisante.

Dès lors le GCL semble plus optimal de par sa rapidité pour un jeu de poker en ligne où des centaines de parties sont jouées simultanément et pendant plusieurs jours mais Blum Blum Shub sera plus efficace dans des parties importantes comme les tournois de poker où il faut s'assurer qu'il n'y aura pas de triche et où moins de parties seront à générer.

Annexes

- **Preuve du lemme :**

Hypothèse : $\exists q \in \mathbb{N}, q \wedge p = 1$ tel que $x = 1 + qp^k$

$$x^p = \sum_{i=0}^p \binom{p}{i} q^i p^{ki} = 1 + q^p p^{kp} + qp^k \sum_{i=1}^{p-1} \binom{p}{i} q^{i-1} p^{k(i-1)}$$

or $p \mid \binom{p}{i} \forall i \in \llbracket 1 ; p-1 \rrbracket$ donc $\exists s_i \in \mathbb{N}, \binom{p}{i} = ps_i$

$$\text{dès lors : } x^p = 1 + qp^{k+1}(q^{p-1}p^{k(p-1)-1}) + qp^{k+1} \sum_{i=1}^{p-1} s_i q^{i-1} p^{k(i-1)}$$

$$\text{donc : } x^p \equiv 1[qp^{k+1}] \text{ et } q \wedge p = 1 \implies x^p \equiv 1[p^{k+1}]$$

$$\text{et } x^p = 1 + qp^{k+2}(q^{p-1}p^{k(p-1)-2}) + qp^{k+1}s_1 + qp^{k+2} \sum_{i=2}^{p-1} s_i q^{i-1} p^{k(i-1)-1}$$

$$\text{or } s_1 = 1 \implies x^p \equiv 1 + qp^{k+1}[qp^{k+2}] \implies x^p \not\equiv 1[p^{k+2}]$$

- **Tableau de chi-carré :**

Degrés de liberté	valeurs limites pour $\alpha = 0.05$
1	3.841
2	5.991
3	7.815
4	9.488
5	11.070
6	12.592
7	14.067
8	15.507
9	16.919
D	$D + 1.64\sqrt{2D} + \frac{2}{3}(1.64^2 - 1) + O(\frac{1}{\sqrt{D}})$

- **Tableau du jeu de Poker :**

Main	Jeux	Probabilité (%)	Jeux / 2M coups
Quinte flush royale	4	0.000154	3.08
Quinte flush	36	0.00139	27.7
Carré	624	0.024	480
Full	3 744	0.144	2 881
Couleur	5 108	0.197	3 930
Quinte	10 200	0.392	7 849
Brelan	54 912	2.133	42 256
Deux paires	123 552	4.754	95 078
Paire	1 098 240	42.257	845 138
Carte haute	1 302 540	50.118	1 002 354
Total	2 598 960	100	2 000 000

Bibliographie

- The Middle Square Method
- The Art of Computer programming (Vol. 2) - D. Knuth
- A simple unpredictable pseudo-random number generator - L.Blum, M.Blum, M.Shub
- Sécurité du générateur de Blum Blum Shub - R. Rolland
- Tests de générateurs pseudo-aléatoires - R.Devillers, J.J. Dumont, G.Latouche
- Testing Number Generators - R.Jain
- Tests du Khi-Deux - L.Houde