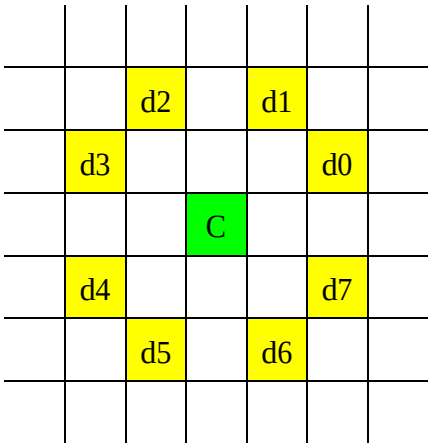
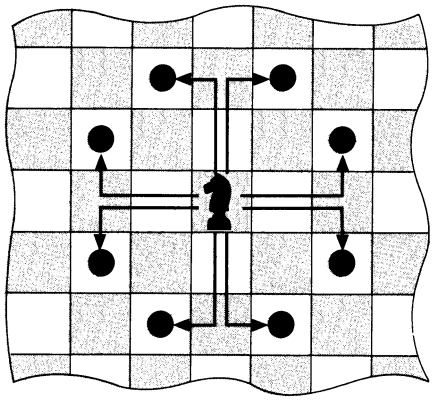
La marche du cavalier

I. POSITION DU PROBLÈME

On désire effectuer un parcours complet de l'échiquier 8×8 avec le cavalier sans repasser deux fois sur la même case.

- *Marche du cavalier*





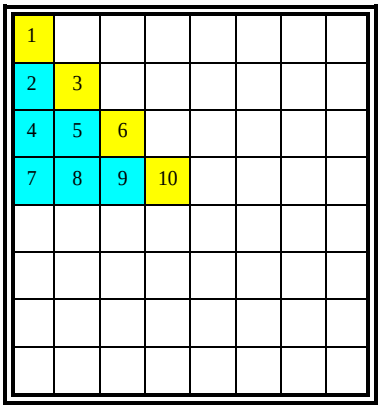
(Ian Stewart, Pour la Science, Septembre 1987, p88)

Le déplacement du cavalier s'effectue par sauts Δx, $\Delta y = \pm 1, \pm 2$. On remarque que tous les déplacements s'effectuent de telle sorte que $\Delta x + \Delta y = -3, -1, 1, 3$ soit $\Delta x + \Delta y = 1 \bmod 2$. On distingue donc deux familles de cases sur l'échiquier : les cases noires et les cases blanches. Le cavalier change de couleur à chaque saut.

- symétries

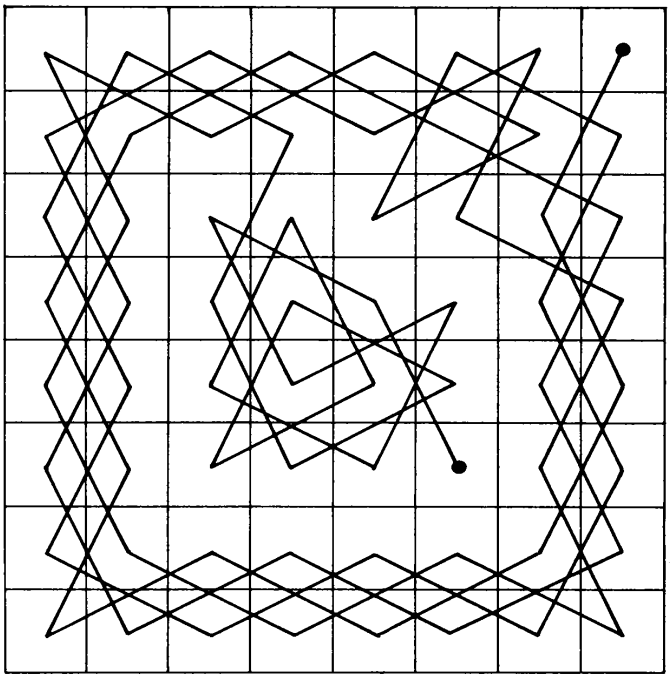
Symétries d'ordre 8 pour le carré et les déplacements, on étudie les seules cases numérotées de 1 à 10. La recherche à partir des cases triangles 2, 4, 5, 7, 8, et 9, conduit à obtenir deux solutions symétriques différentes, contrairement à ce qui se passe avec les cases diagonales 1, 3, 6, et 10.
De plus une solution obtenue est réversible, c'est à dire que le cavalier peut effectuer le trajet dans les deux sens (2 solutions distinctes).
Notons aussi qu'un parcours fermé peut être décrit à partir de n'importe quelle case (64 solutions identiques).

- Dessin de l'échiquier

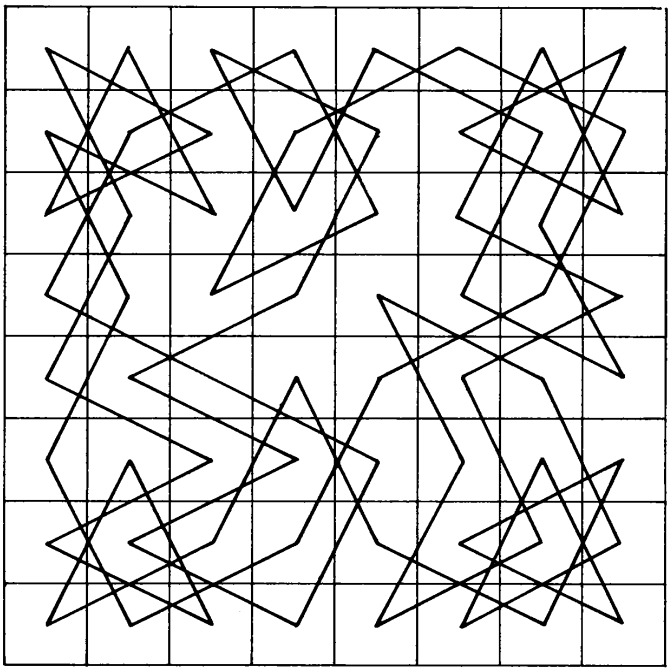


- solutions proposées (Ian Stewart, Pour la Science, Septembre 1987, p88)

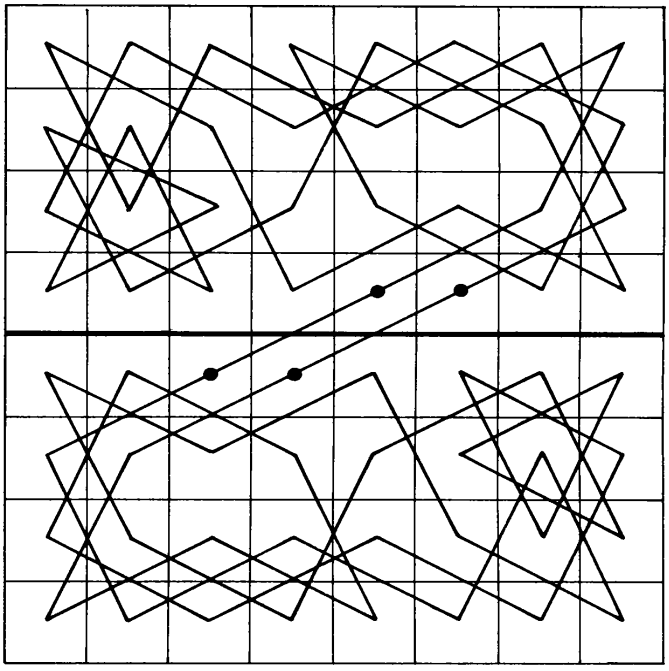
solution proposée par Abraham De Moivre



solution proposée par Adrien-Marie Legendre



solution proposée par Léonhard Euler



II. ALGORITHME D'OPTIMISATION D'EULER

Algorithme d'optimisation. Il faut tester chacune des cases où il est possible de se rendre. Éviter de sortir du cadre. Éviter de considérer une case déjà visitée. On choisit la case qui offre le moins de possibilités ultérieures de déplacement. Cette stratégie conduit à "suivre les bords", puisqu'ils offrent naturellement moins de possibilités de déplacements que les cases centrales.

1		3				13	
4				14			
	2	p6				p4	12
	5		p6		p6		
						11	
6							
			8				10
	7				9		

C'est un algorithme très efficace qui produit des solutions avec une forte probabilité de succès (> 91 %). On utilise la récursivité et le "backtraking" (retour en arrière) pour obtenir toutes les solutions. Par exemple :

1	16	41	22	3	18	61	48
40	23	2	17	60	49	4	19
15	42	39	56	21	62	47	64
24	37	44	59	50	57	20	5
43	14	51	38	55	46	63	30
36	25	54	45	58	31	6	9
13	52	27	34	11	8	29	32
26	35	12	53	28	33	10	7

Nombre d'embranchements rencontrés dans l'arbre de recherche :

```
122111112111111111211112112122112211111123112311232111221121111
```

Numéros des branches explorées pour obtenir ce parcours :

```
112111112111111111111111111121112111111122112311131111121111111
```

On constate que le nombre de possibilités d'embranchements est relativement réduit. Il est possible de parcourir l'arbre complet en quelques minutes.

Le bilan complet des solutions obtenues par l'algorithme d'Euler est le suivant :

Trajets réussis : **7894584**

137300			
205491	69388		
14570	84653	60488	
52469	148423	92424	510410

soit $4 \times D + 8 \times T = 7894584$ solutions distinctes

Trajets fermés réussis : **1188384**

2662			
13346	6854		
1288	13604	15706	
4622	33991	19107	99958

soit $4 \times D + 8 \times T = 1188384$ solutions distinctes

Échecs de parcours : **761520**

6804			
13725	6972		
786	6958	6404	
3587	16452	10431	66322

soit $4 \times D + 8 \times T = 761520$ échecs

Les parcours fermés représentent 15 % des solutions trouvées. Les échecs dans la recherche ne représentent que 9 % des tentatives. Cette méthode de recherche est donc assez efficace pour construire une solution "à la main".

III. RECHERCHE DE SOLUTIONS DIFFÉRENTES

On constate que le nombre de possibilités d'embranchements dans l'arbre de recherche, évalué selon la méthode précédente, est relativement réduit. Il doit être possible de construire des solutions avec un plus grand nombre d'embranchements dans les premiers noeuds de l'arbre.

Combien y a-t-il d'embranchement possibles ? Quelle est la taille de l'arbre complet ? Combien y a-t-il de solutions ? Combien de parcours fermés ?

Première approche

- Parcours de l'arbre complet

On simplifie le programme précédent en supprimant la règle d'optimisation d'Euler. On explore alors récursivement toutes les branches de l'arbre, en prenant en compte tous les coups possibles à un certain niveau. Cet algorithme rapide permet de tester environ 6×10^{10} parcours par jours (processeur Athlon 650 MHz).

1	24	9	62	45	26	41	60
10	63	2	25	40	61	46	27
23	8	11	44	21	30	59	42
64	3	22	39	58	43	28	47
7	12	5	20	29	48	31	56
4	17	14	53	38	57	34	49
13	6	19	16	51	36	55	32
18	15	52	37	54	33	50	35

Nombre d'embranchements rencontrés dans l'arbre de recherche :

```
2553733331752514513465113224455224131323123311151211221212111111
```

Numéros des branches explorées pour obtenir ce parcours :

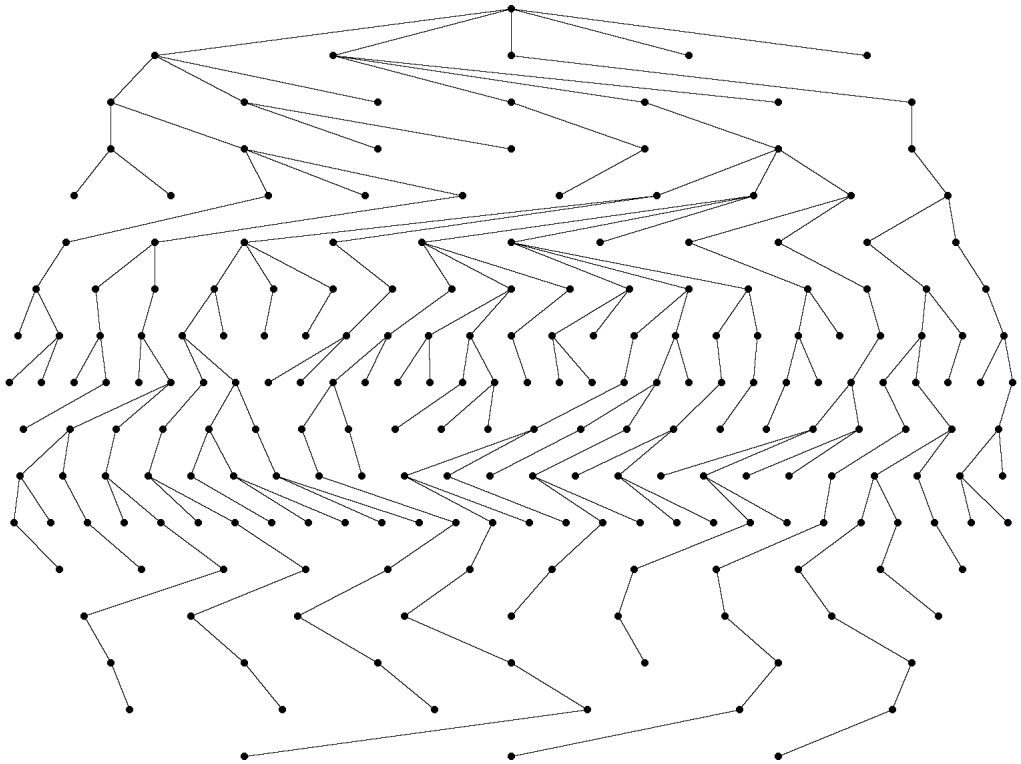
```
1111111111111111111111111112255124111111111211141111221112111111
Solutions : 777                          Nombre d'essais : 585340667
```

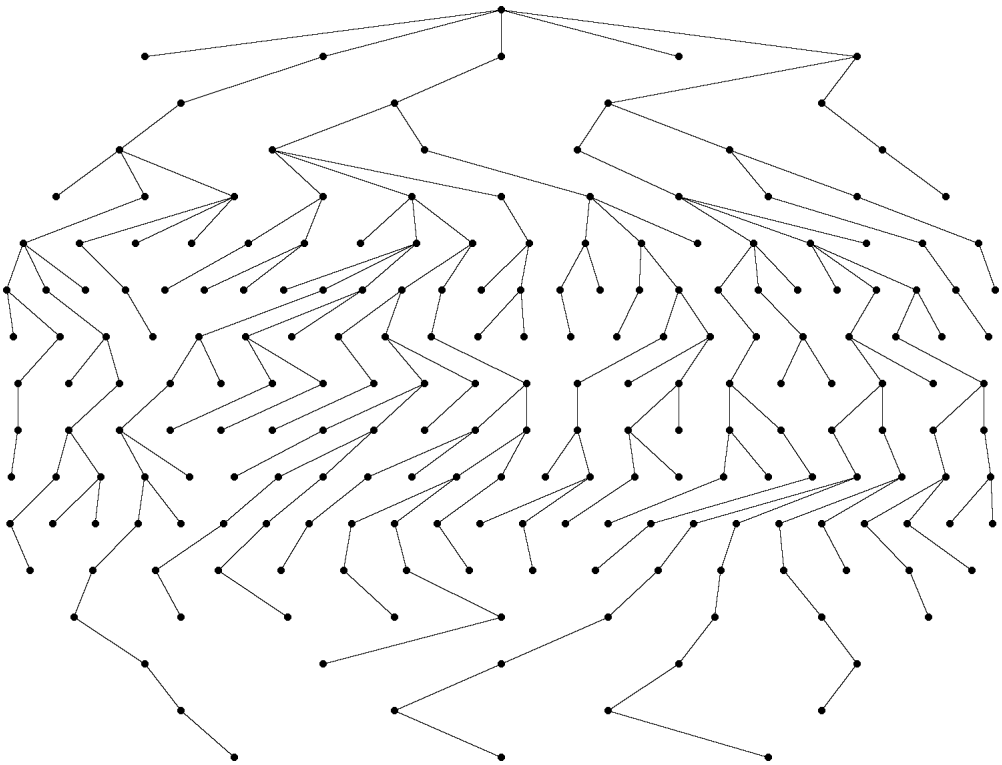
En observant l'évolution de la liste des noeuds explorés, on constate néanmoins que cette méthode est trop lente. Le nombre de possibilités de bifurcation est relativement important. Sur cet exemple il est de l'ordre de 2.6×10^{21}.

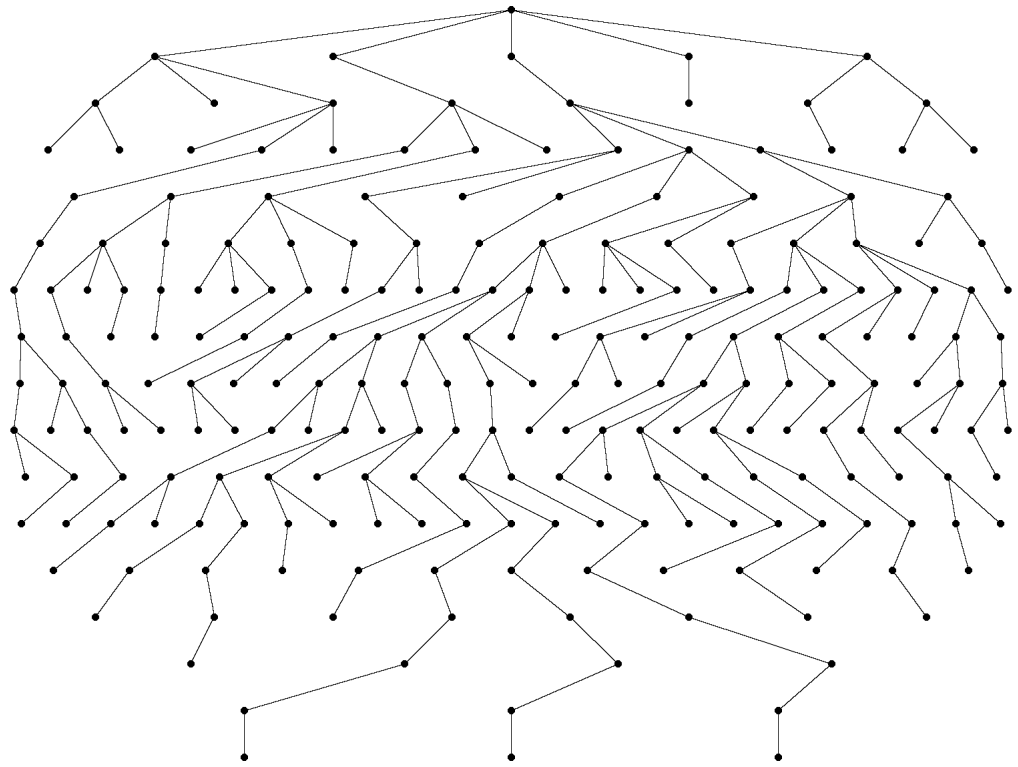
A partir du point de départ (1, 1) on obtient, en 3 heures de calculs, 2500 solutions qui présentent les mêmes 24 premier coups, et les mêmes 5 derniers. A partir du point de départ (4, 4) on n'obtient aucune solution dans le même délai.

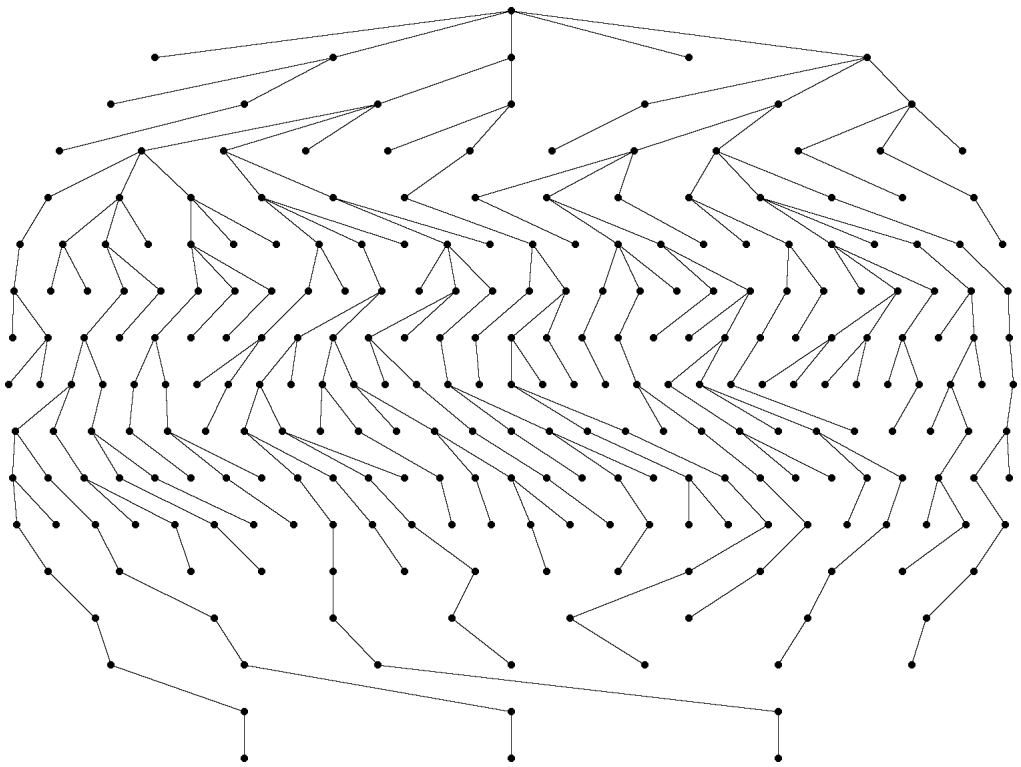
L'arbre semble finalement beaucoup trop grand. Si on se trouve dans une branche stérile (dont la longueur maximale est inférieure à 63 sauts), le risque est grand de ne jamais trouver de solution. Il est donc très difficile de se faire une opinion sur le nombre de solutions en utilisant cette méthode.

- Dessins de l'arbre de recherche (parcours aléatoires et chemins hamiltoniens)

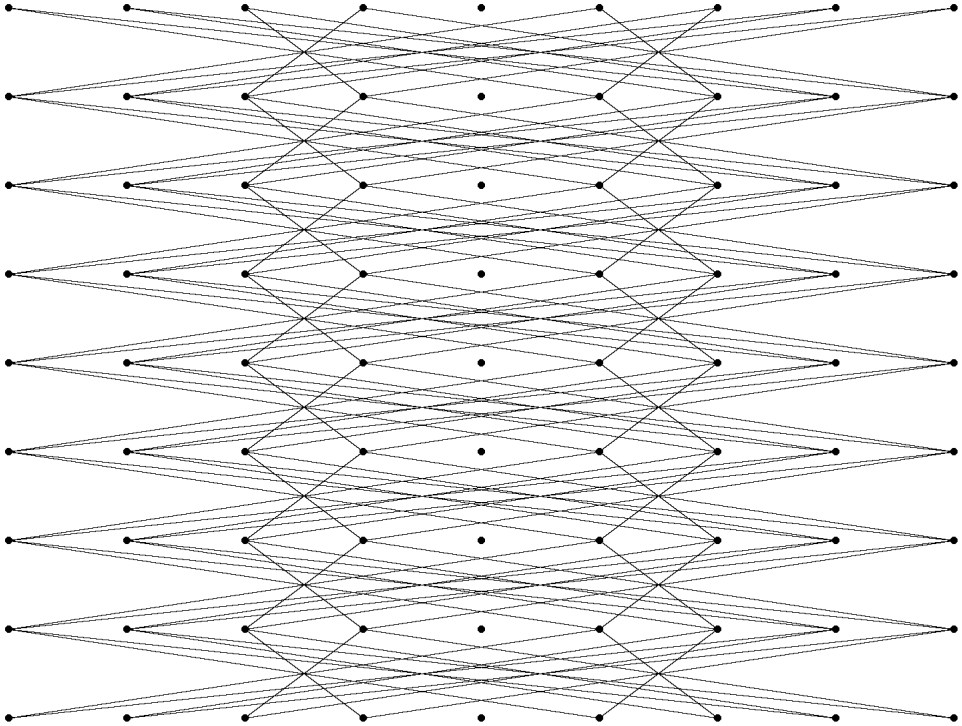


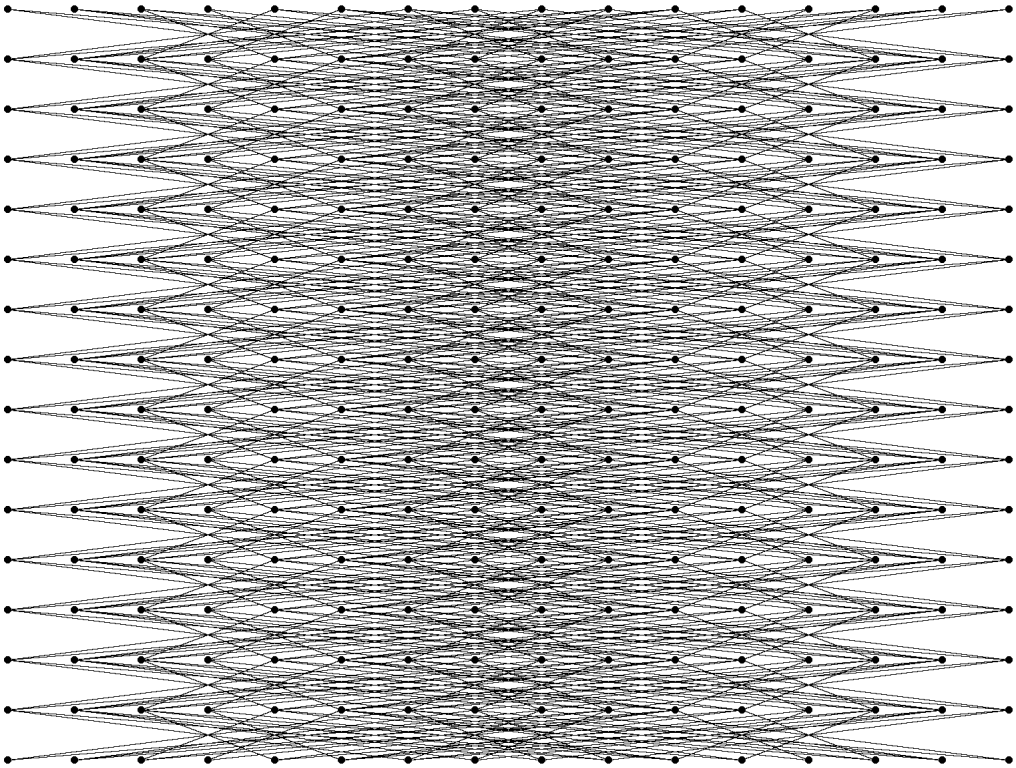


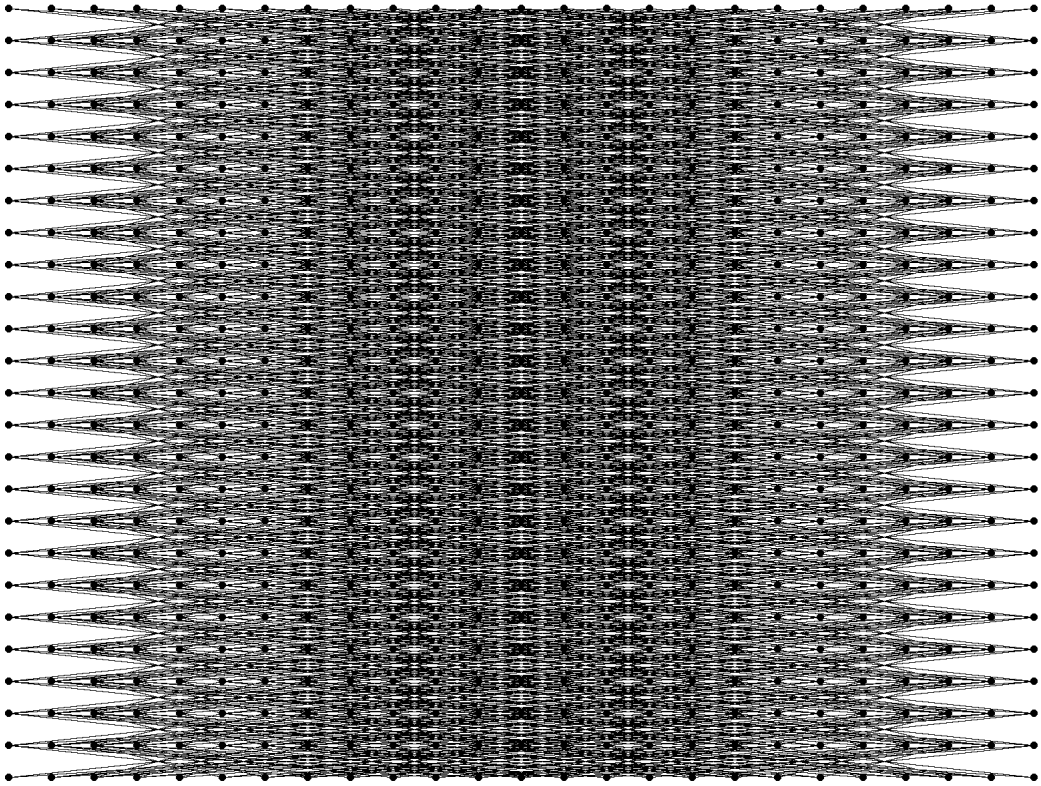




Chemins hamiltoniens dans les échiquiers 3´3, 4´4 et 5´5







- Parcours de l'arbre par recherche pseudo-aléatoire de chemin

Dans la mesure où le calcul complet est trop difficile, il semble judicieux d'utiliser un parcours aléatoire des embranchements, pour évaluer le nombre effectif de solutions. En considérant une case de départ donnée, 63 paramètres (au maximum) définissent un parcours dans l'arbre, mais tous les trajet n'ont pas une longueur de 63 sauts, et tous les noeuds ne présentent pas 8 possibilités. Il y a donc sûrement beaucoup moins de $8^{63} \approx 7.8 \times 10^{56}$ solutions.

Amélioration de la valeur du majorant, en considérant les 63 possibilités de saut :

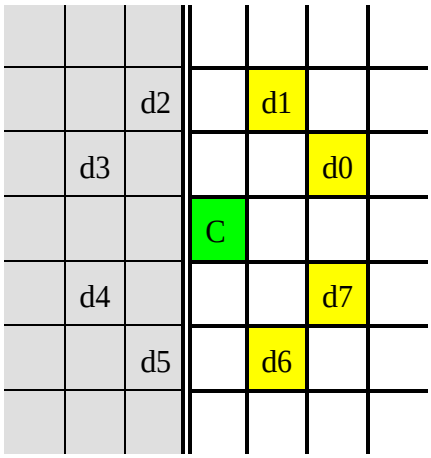
2	3	4	4
3	4	6	6
4	6	8	8
4	6	8	8

soit au maximum $[2^{27}3^6]^4/2 = 2^{107}3^{24} \approx 4.6 \times 10^{43}$ solutions.

On utilise donc maintenant une méthode de Monte-Carlo. La recherche s'effectue à partir de la case initiale, pour chacun des essais réalisés. Le traitement de chaque parcours est plus long que dans la méthode de recherche systématique, puisqu'alors le programme explorait de nombreuses branches sans retour à la racine.

On reprend le programme récursif précédent mais on choisit chaque déplacement à l'aide d'un tirage aléatoire. Tant que l'avance du cavalier est possible, on effectue un nouveau tirage *x* (compris dans l'intervalle [0, 7]). On explore la liste des cases accessibles à partir de la valeur de *x*, $X_0 = x$, $X_n = (X_{n-1} + 1) \bmod 8$, suivant l'ordre des déplacements définit en introduction, jusqu'à ce que la case X_n soit une case libre.

Première difficulté : les solutions obtenues ressemblent aux précédentes, "en suivant les bords". Ceci est lié à la façon de tirer dans la liste des cases accessibles, à partir d'une valeur *x*, puis en tournant dans le sens direct. En fait, il faut créer la liste des cases accessibles, et ne tirer au sort que parmi celles-ci.



Deuxième difficulté : les solutions sont si rares qu'il faut maintenant réaliser environ 6×10^6 essais pour en obtenir une. Et chaque tentative nécessite le tirage de 35 nombres pseudo-aléatoires, en moyenne. Soit 2×10^8 tirages aléatoires par solution trouvée. Le générateur 32 bits (2^{32} états) du Turbo Pascal ne permet de déterminer qu'une vingtaine de solutions indépendantes.

Troisième difficulté : la synchronisation des tirages entre deux tours successifs se réalise, même si la séquence pseudo-aléatoire ne démarre pas de la même manière. On ne peut espérer bénéficier d'un décalage de la séquence pseudo-aléatoire entre deux tours consécutifs.

Les processeurs récents permettent de réaliser près de 10^{11} tirages par jour (soit un tirage toutes les microsecondes), il est donc nécessaire de disposer d'un générateur de plus longue période. On obtient alors environ 500 solutions indépendantes par 24 heures de calcul, soit une nouvelle solution toute les 3 minutes.

Astuces de réalisation : Utilisation d'un carré 8×8 entouré d'une protection de deux rangées pour éviter les tests de sortie de l'échiquier, soit finalement gestion d'un tableau carré 12×12. Utilisation d'un tableau à une dimension, plus rapide que la gestion d'un tableau à deux dimensions, avec déplacements du cavalier respectivement égaux à -10, -23, -25, -14, 10, 23, 25 et 14.

IV. GÉNÉRATEURS PSEUDO-ALÉATOIRES

Différentes méthodes de génération peuvent être utilisées. Les deux caractéristiques essentielles sont un bon caractère aléatoire et la rapidité d'évaluation de la séquence.

Tous les tests de générateurs sont réalisés avec des valeurs de X_n réelles, comprises dans l'intervalle $]0, 1[$, valeurs qu'on obtient en normalisant par un majorant de la suite. Pour le parcours de l'échiquier, on obtient un entier compris dans l'intervalle $[0, 7]$ en multipliant le réel précédent par 8, et en calculant la partie entière.

- Méthode de congruence linéaire (D. Knuth, tome 2, page 10)

Introduite par Lehmer en 1949, c'est la méthode la plus utilisée aujourd'hui. La séquence pseudo-aléatoire $<X_n>$ est obtenue par la formule :

$$X_{n+1} = (aX_n + c) \bmod m$$

pour laquelle il faut choisir quatre nombres :

m,	le module
a,	le multiplicateur
c,	l'incrément
X_0,	la valeur initiale ou graine

C'est le choix de ces quatre nombres qui détermine la qualité pseudo-aléatoire de la séquence générée. Les valeurs utilisées sont celles du générateur de Maple, pour lequel :

$$m = 999999999989 = 10^{12} - 11,\ a = 427419669081,\ c = 0.$$

Dans le cas du Turbo Pascal et de Delphi les valeurs utilisées sont les suivantes :

$$m = 2^{32},\ a = 134775813 = 08088405\text{h} = 3 \times 17 \times 131 \times 20173,\ c = 1.$$

La séquence est évaluée à l'aide de multiplication, addition et division. On verra que l'on a quelques difficultés avec la précision des calculs.

- Polynômes irréductibles (Lin Costello, Error Control Coding GF(2^n))

On utilise les polynômes irréductibles sur le corps de Galois GF(2^n) des codes BCH (Bose, Chaudhuri, et Hocquenghem), et les racines primitives de ces polynômes.

Par exemple avec $p(X) = 1 + X + X^4$ sur GF(2^4), on obtient la séquence suivante de période $2^4 - 1$:

1	1														
2		α													
4			α^2												
8				α^3											
3	1	+	α												
6		α	+	α^2											
12			α^2	+	α^3										
11	1	+	α		+	α^3									
5	1			+	α^2										
10		α		+	α^3										
7	1	+	α	+	α^2										
14		α	+	α^2	+	α^3									
15	1	+	α	+	α^2	+	α^3								
13	1			+	α^2	+	α^3								
9	1					+	α^3								

La séquence est évaluée facilement, à l'aide de décalages et de ou exclusif. J'ai effectué des tests avec les valeurs suivantes des polynômes irréductibles :

P[0] = *X*^39 + *X*^38 + *X*^37 + *X*^36 + *X*^34 + *X*^33 + *X*^31 + *X*^28 + *X*^27 + *X*^25 + *X*^23 + *X*^22 + *X*^17 + *X*^11 + *X*^8 + *X*^5 + 1

P[1] = *X*^45 + *X*^43 + *X*^42 + *X*^41 + *X*^40 + *X*^37 + *X*^36 + *X*^31 + *X*^29 + *X*^28 + *X*^26 + *X*^24 + *X*^21 + *X*^19 + *X*^16 + *X*^15 + *X*^14 + *X*^12 + *X*^9 + *X*^8 + *X*^7 + *X*^6 + *X*^4 + *X*^2 + 1

P[2] = *X*^47 + *X*^46 + *X*^43 + *X*^42 + *X*^40 + *X*^39 + *X*^36 + *X*^33 + *X*^32 + *X*^27 + *X*^25 + *X*^24 + *X*^23 + *X*^22 + *X*^20 + *X*^19 + *X*^18 + *X*^16 + *X*^13 + *X*^12 + *X*^11 + *X*^9 + *X*^8 + *X*^5 + *X*^3 + *X*^1 + 1

P[3] = *X*^53 + *X*^50 + *X*^49 + *X*^48 + *X*^46 + *X*^44 + *X*^43 + *X*^40 + *X*^37 + *X*^34 + *X*^33 + *X*^29 + *X*^27 + *X*^26 + *X*^24 + *X*^22 + *X*^20 + *X*^19 + *X*^18 + *X*^16 + *X*^14 + *X*^12 + *X*^6 + *X*^5 + *X*^3 + *X*^2 + 1

P[4] = *X*^56 + *X*^54 + *X*^52 + *X*^49 + *X*^48 + *X*^45 + *X*^41 + *X*^38 + *X*^36 + *X*^35 + *X*^33 + *X*^32 + *X*^28 + *X*^27 + *X*^26 + *X*^24 + *X*^19 + *X*^18 + *X*^16 + *X*^14 + *X*^13 + *X*^12 + *X*^9 + *X*^8 + *X*^7 + *X*^6 + *X*^4 + *X*^3 + *X*^2 + *X*^1 + 1

Les générateurs obtenus ne sont pas de bonne qualité. Mais une erreur de programmation m'a permis de réaliser avec P[1] un générateur qui passait les test réalisés.

- Augmentation de période par addition (D. Knuth, tome 2, page 27)

On utilise une formule du type de la congruence linéaire :

$$X_n = (X_{n-p} + X_{n-q}) \bmod m$$

ce qui permet de produire différents types de générateurs.

Lorsque $p = 1$ et $q = 2$, on obtient les termes de la suite de Fibonacci, mais ceci constitue en fait un bon exemple de mauvais générateur pseudo-aléatoire.

Lorsque $p = 1$ et $q = k$, on obtient un générateur qui fonctionne de manière satisfaisante pour des valeurs de k supérieures ou égales à 16.

Un bien meilleur résultat est obtenu lorsque $p = 24$ et $q = 55$, comme cela a été proposé par Mitchell et Moore en 1959. Si $m = 2^e$, la période de la suite $<X_n>$ est égale à $2^{e-1}(2^{55} - 1)$. J'ai finalement choisit cette dernière suite avec $e = 30$, ce qui conduit à une période voisine de $2^{84} \approx 1.9 \times 10^{25}$.

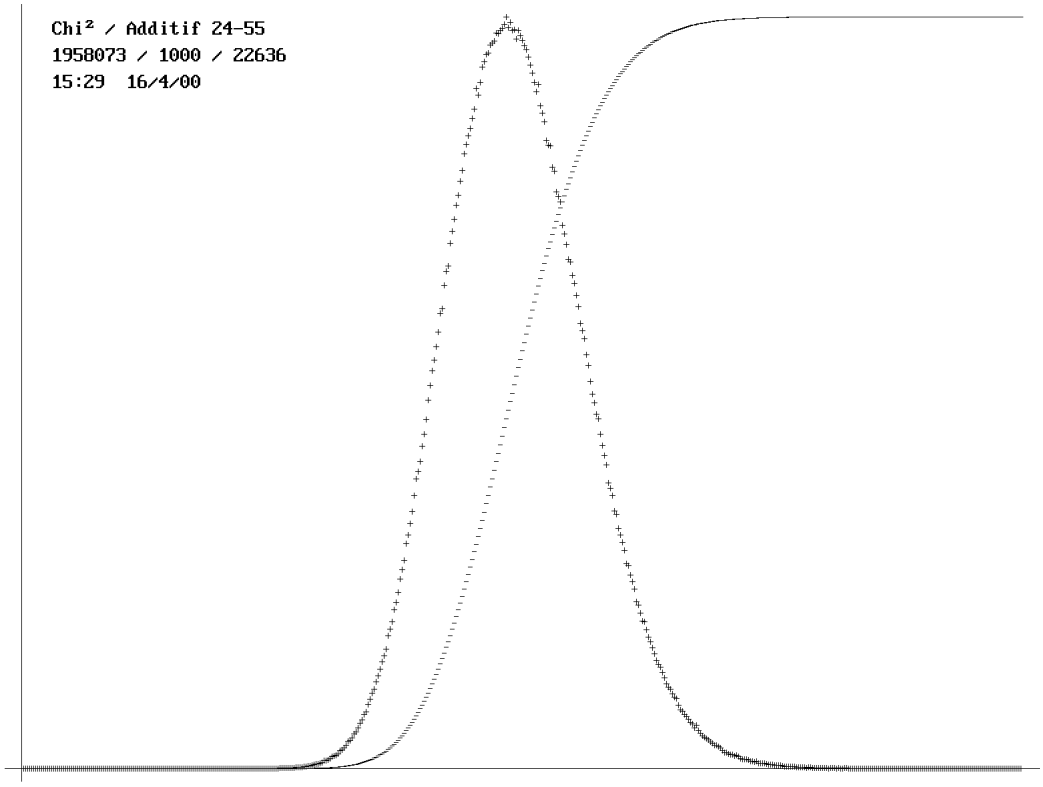
V. TESTS D'UN GÉNÉRATEUR PSEUDO-ALÉATOIRE

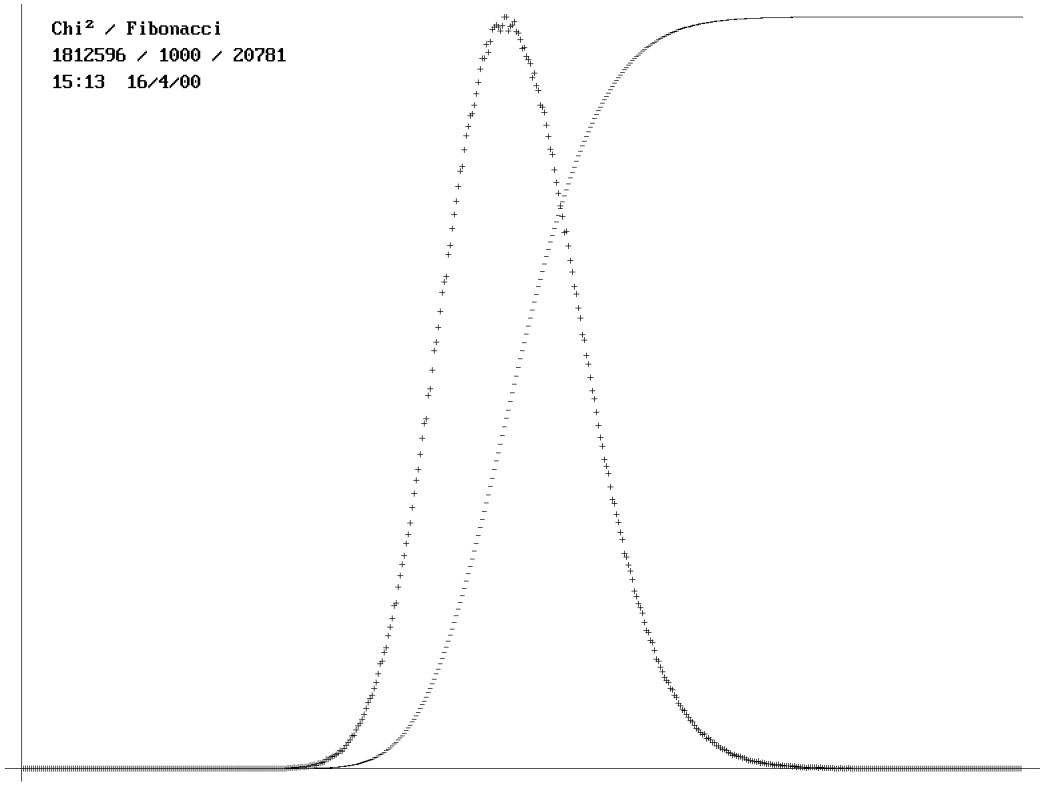
- Test du χ^2 (D. Knuth, tome 2, page 42)

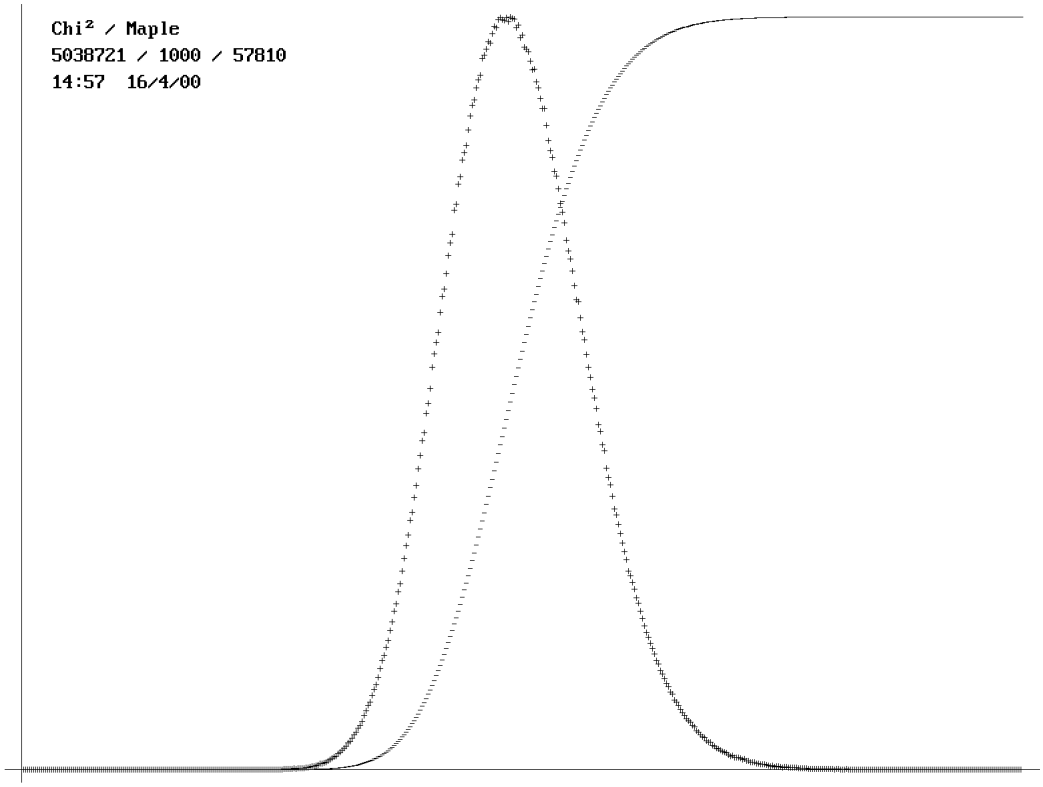
On génère N termes de la suite $<X_n>$ qui sont répartis dans p canaux, numérotés de 0 à $p-1$. On réalise un calcul de χ^2 sur les nombres de fois où le terme X_n a atteint le canal q. La valeur moyenne obtenue dans chaque canal est égale à N/p.

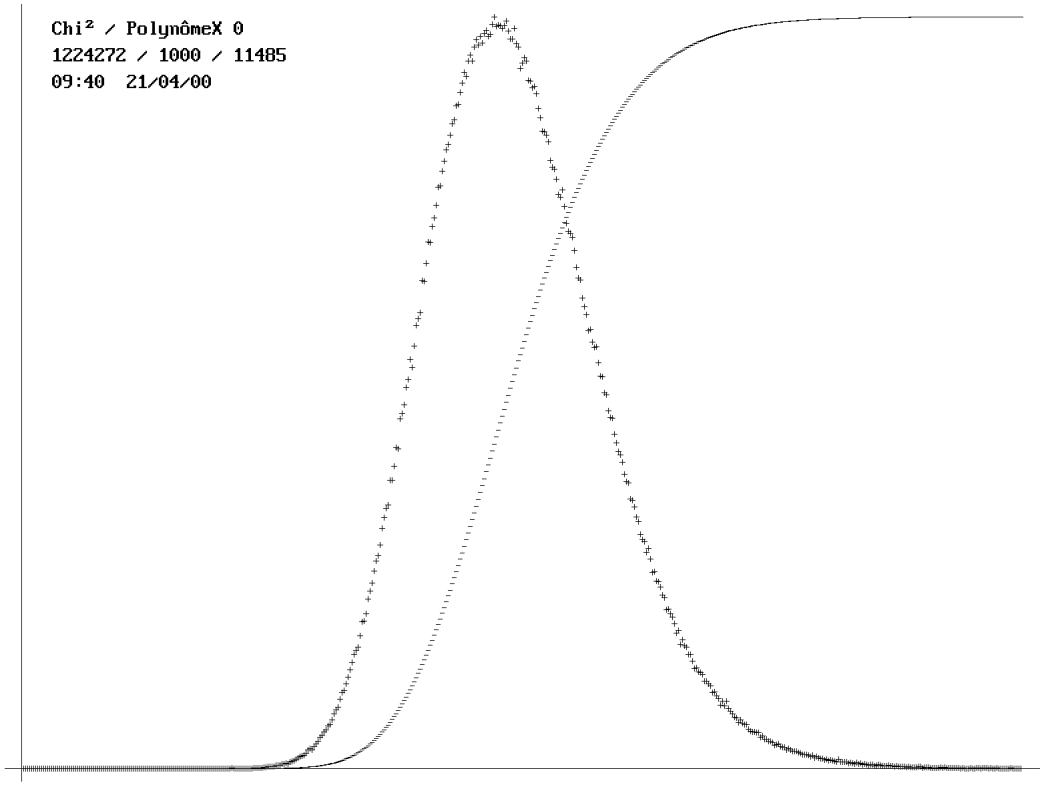
Les tests sont réalisés avec $N = 1000$ et $p = 100$. Les paramètres indiqués sont respectivement le nom du test, la nature du générateur, le nombre d'essais effectués, le nombre N, la hauteur du pic, l'heure et la date de réalisation du test.

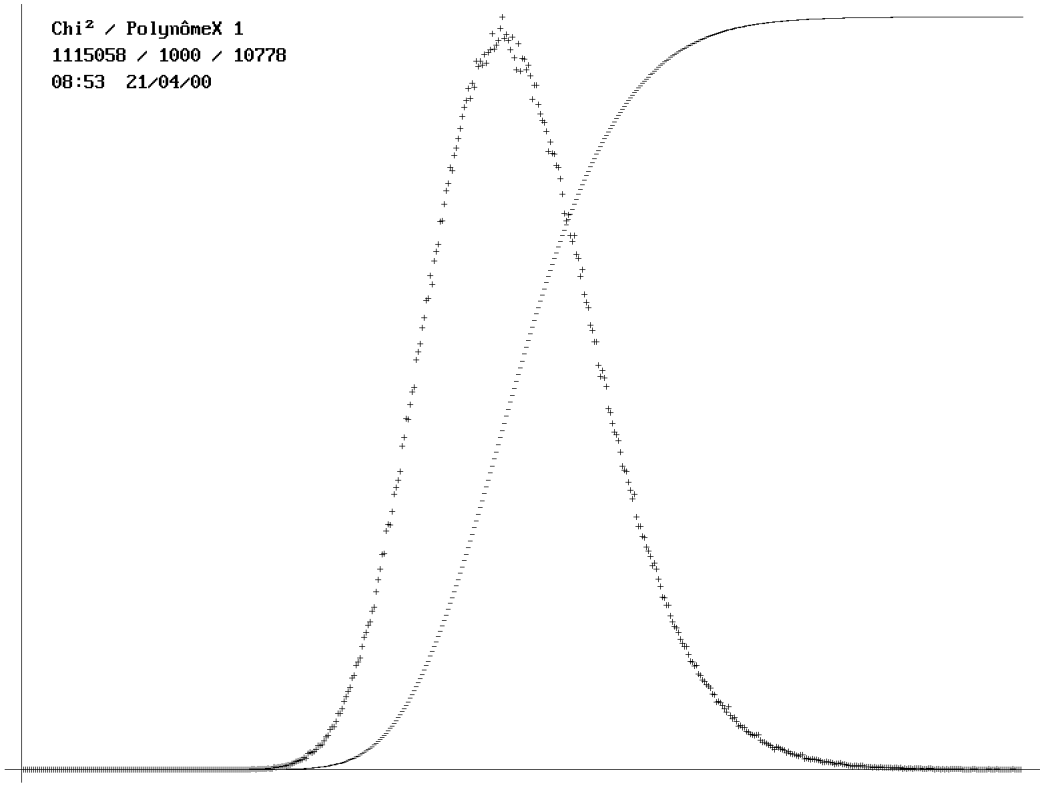
Le test de χ^2 ne permet pas de discerner les générateurs de bonne qualité des mauvais générateurs.

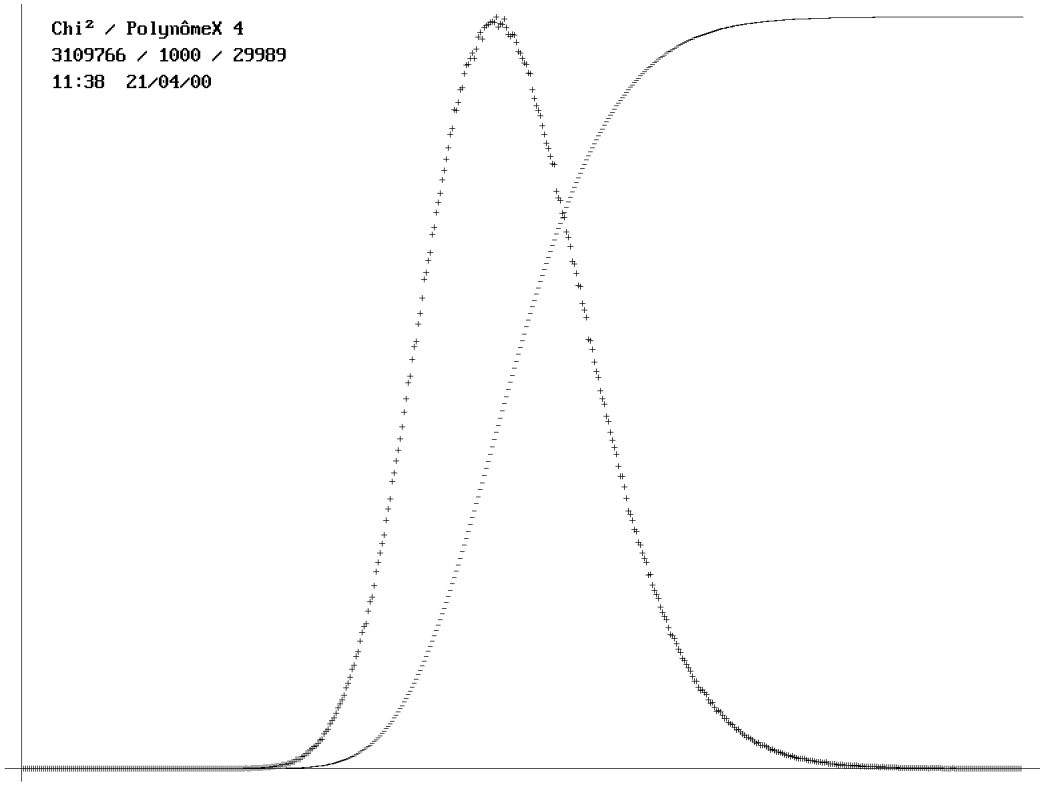



Chi² / Fibonacci
1812596 / 1000 / 20781
15:13 16/4/00


Chi² / Maple
5038721 / 1000 / 57810
14:57 16/4/00


Chi² / PolynômeX 0
1224272 / 1000 / 11485
09:40 21/04/00


Chi² / PolynômeX 1
1115058 / 1000 / 10778
08:53 21/04/00


Chi² / PolynômeX 4
3109766 / 1000 / 29989
11:38 21/04/00

- Test de Kolmogorov Smirnov (D. Knuth, tome 2, page 48)

On considère ici le tirage de p valeurs de la suite $<X_n>$ et la fonction

$$F_p(x) = (\text{ Nombre d'éléments de l'ensemble } X_1 .. X_p \text{ inférieurs à } x) / p$$

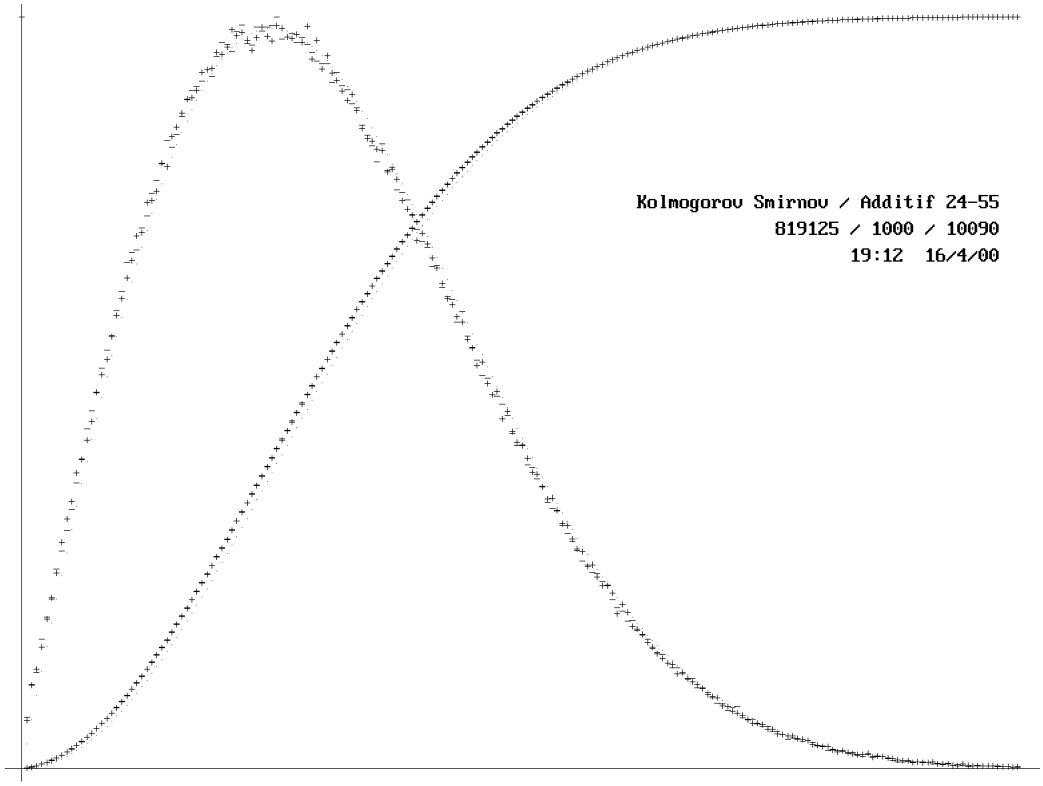
La valeur limite de cette fonction est $F(x) = x$ sur l'intervalle $[0, 1]$, et on calcule l'écart maximal entre les fonctions F_p et F, soit

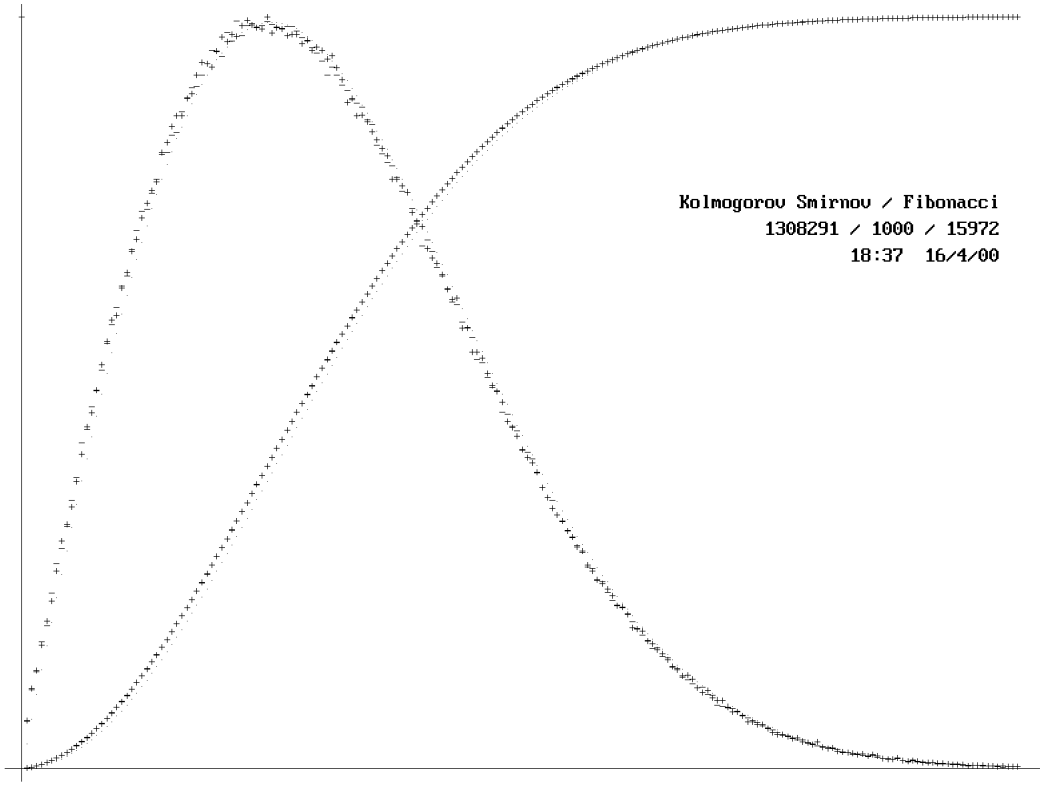
$$K_p^+ = \sqrt{p} \max_{0<x<1} \left(F_p(x) - F(x) \right) \qquad K_p^- = \sqrt{p} \max_{0<x<1} \left(F(x) - F_p(x) \right)$$

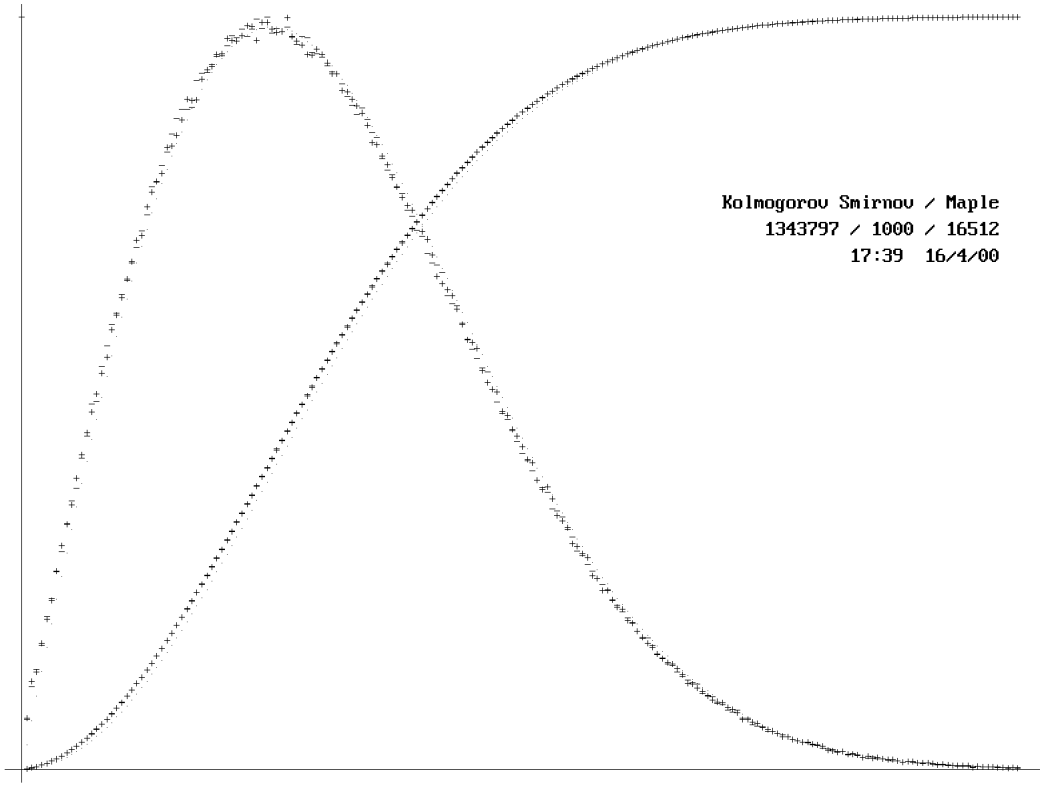
Lorsque p est supérieur à 30 les distributions des valeurs de K_p^+ et de K_p^- sont très bien approchées par la fonction :

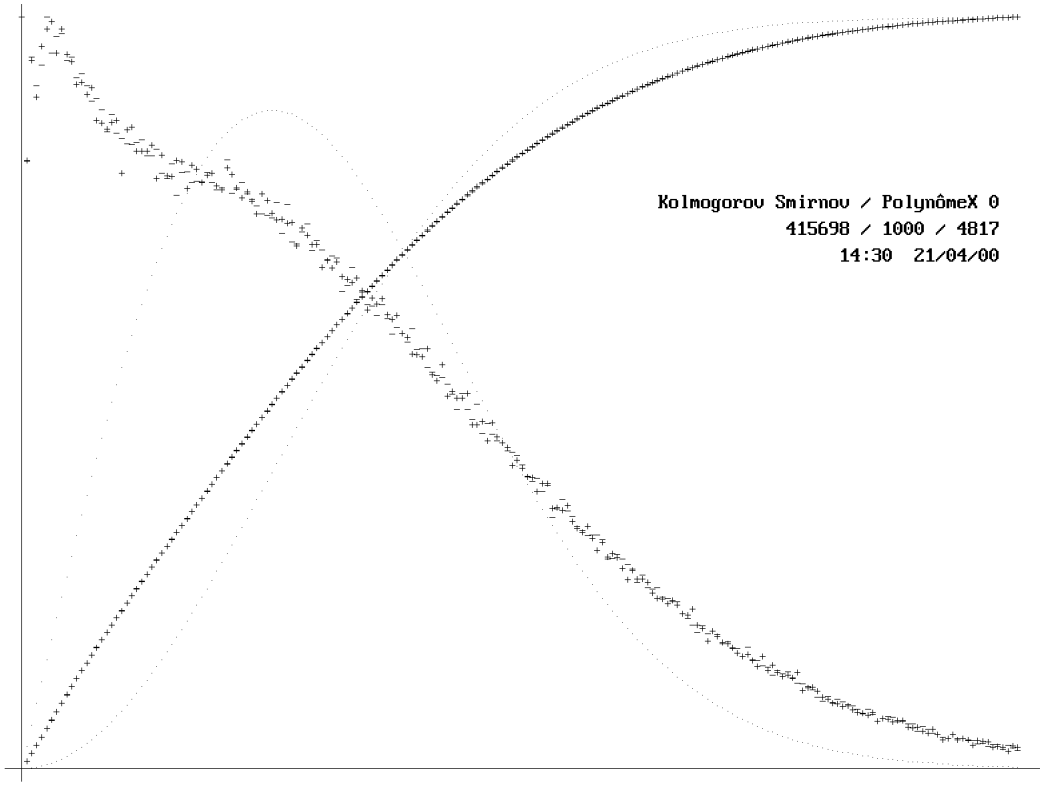
$$G(x) = 1 - e^{-2x^2}$$

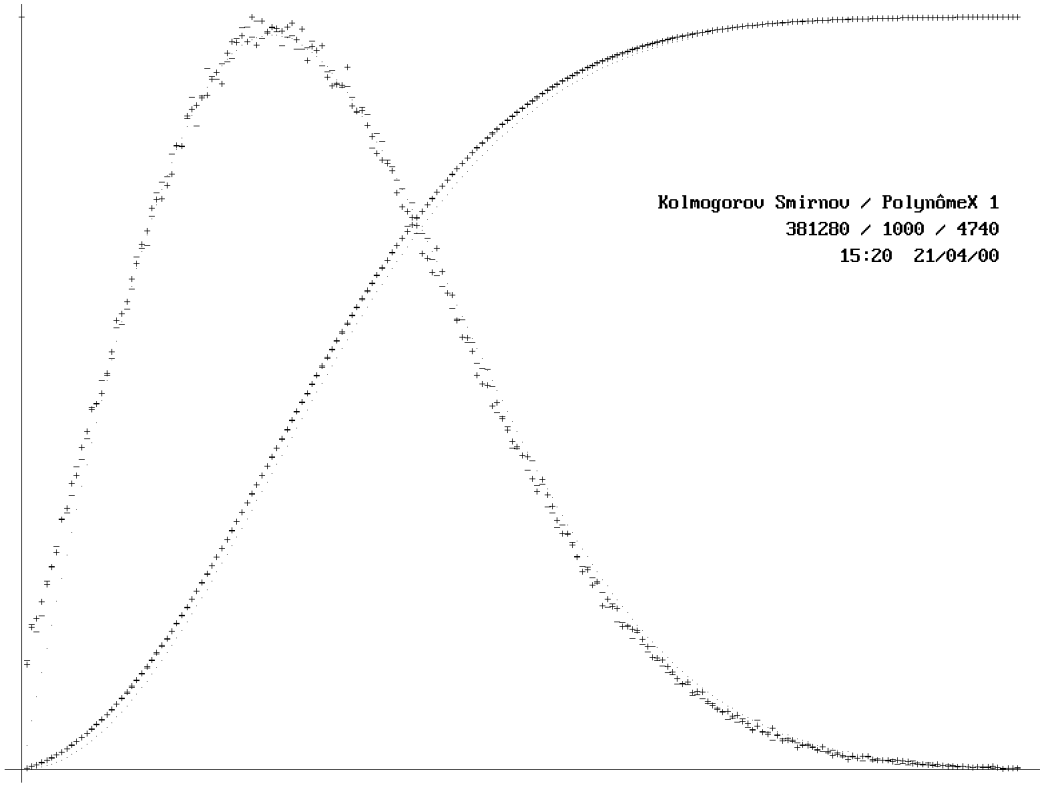
Lors des tests, j'ai utilisée la valeur de $p = 1000$ proposée par D. Knuth, ce qui permet de vérifier la validité des résultats. Les paramètres indiqués sont respectivement le nom du test, la nature du générateur, le nombre d'essais effectués, le nombre p, la hauteur du pic, l'heure et la date de réalisation du test.

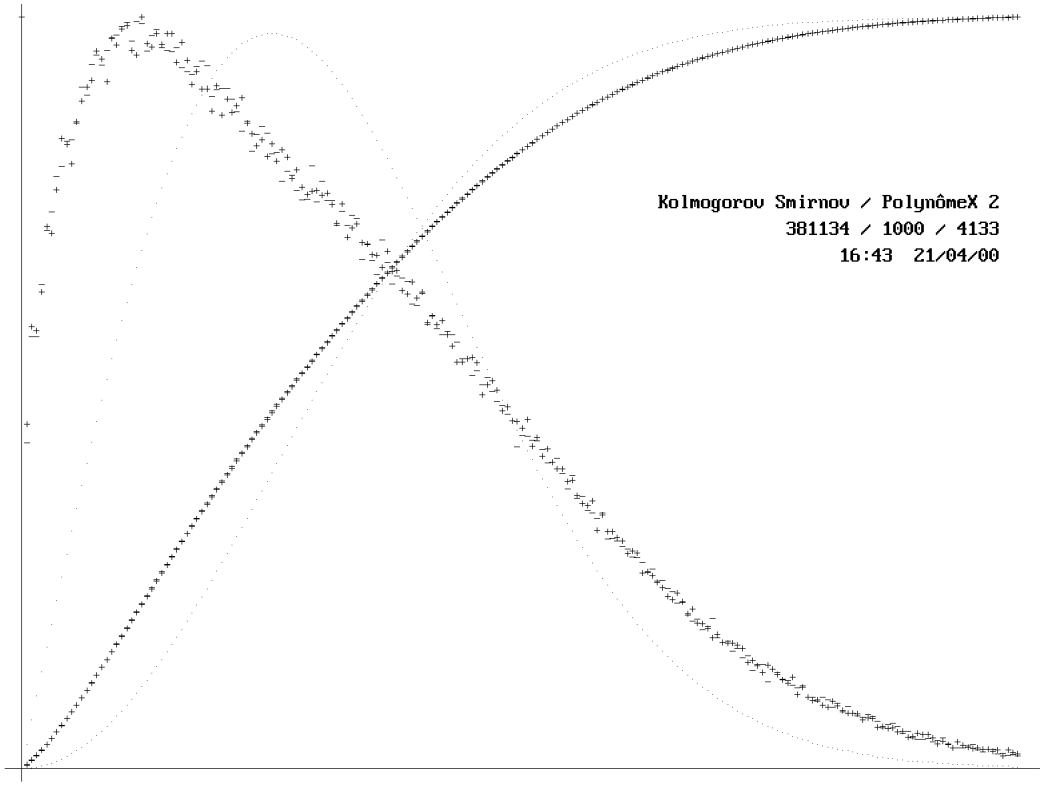


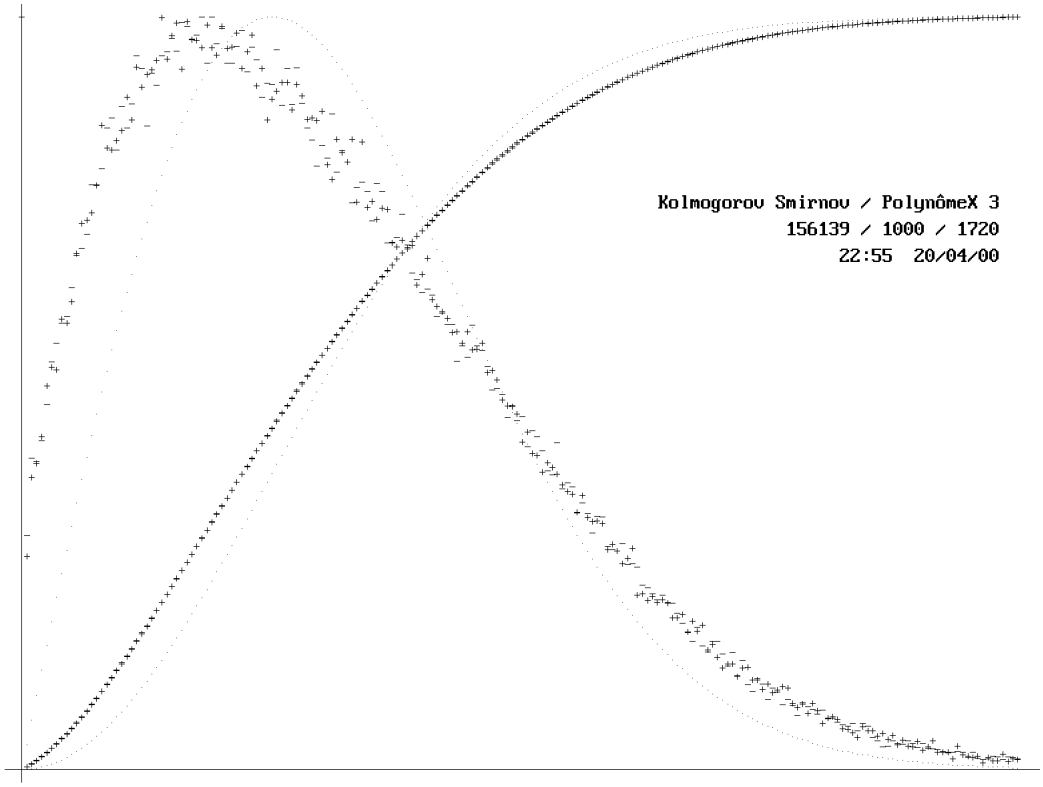

Kolmogorov Smirnov / Fibonacci
1308291 / 1000 / 15972
18:37 16/4/00

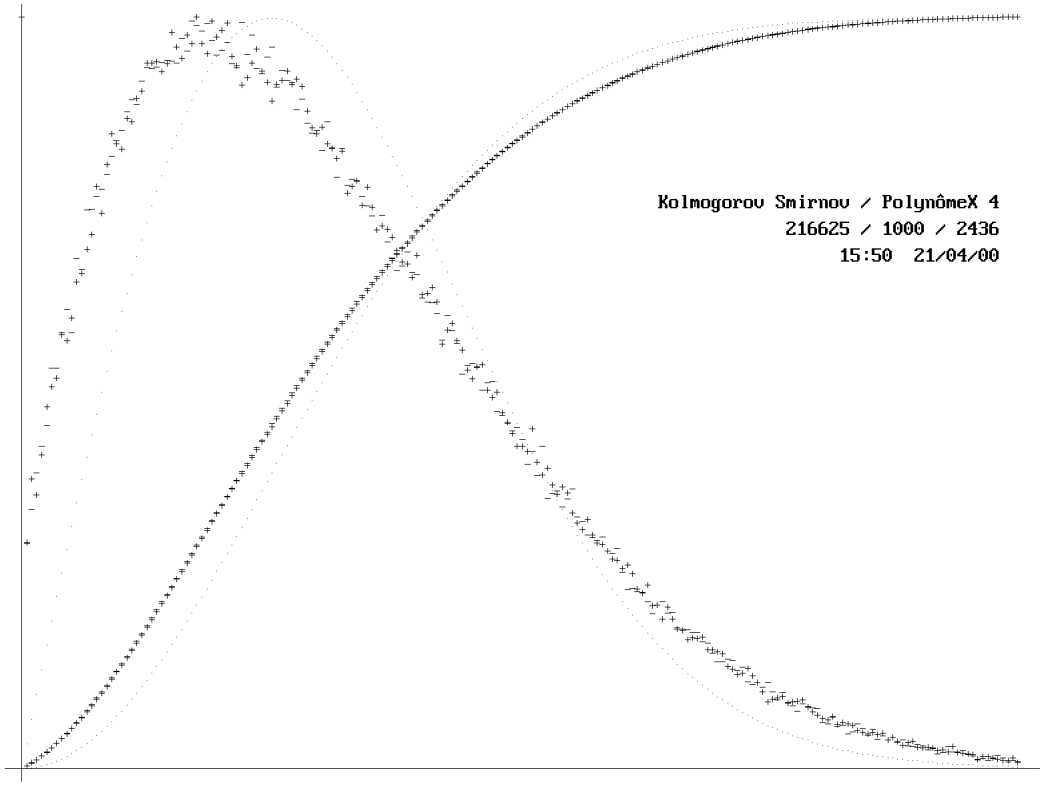

Kolmogorov Smirnov / Maple
1343797 / 1000 / 16512
17:39 16/4/00

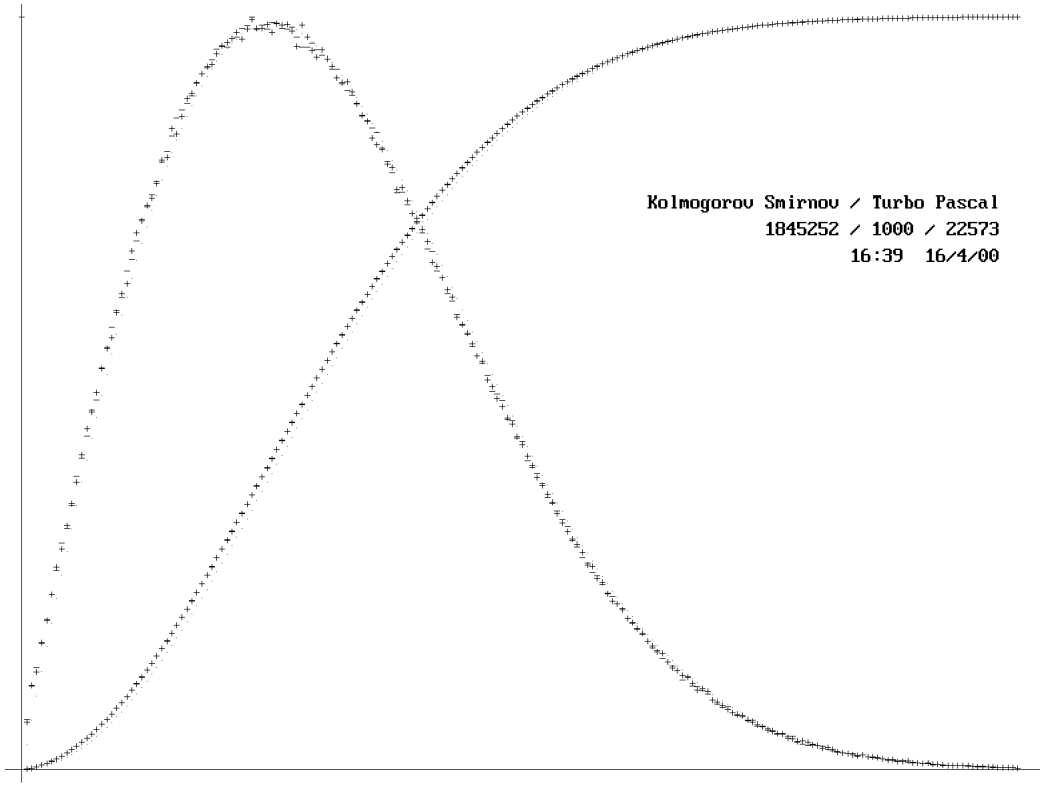

Kolmogorov Smirnov / PolynômeX 0
415698 / 1000 / 4817
14:30 21/04/00


Kolmogorov Smirnov / PolynômeX 1
381280 / 1000 / 4740
15:20 21/04/00


Kolmogorov Smirnov / PolynômeX 2
381134 / 1000 / 4133
16:43 21/04/00


Kolmogorov Smirnov / PolynômeX 3
156139 / 1000 / 1720
22:55 20/04/00

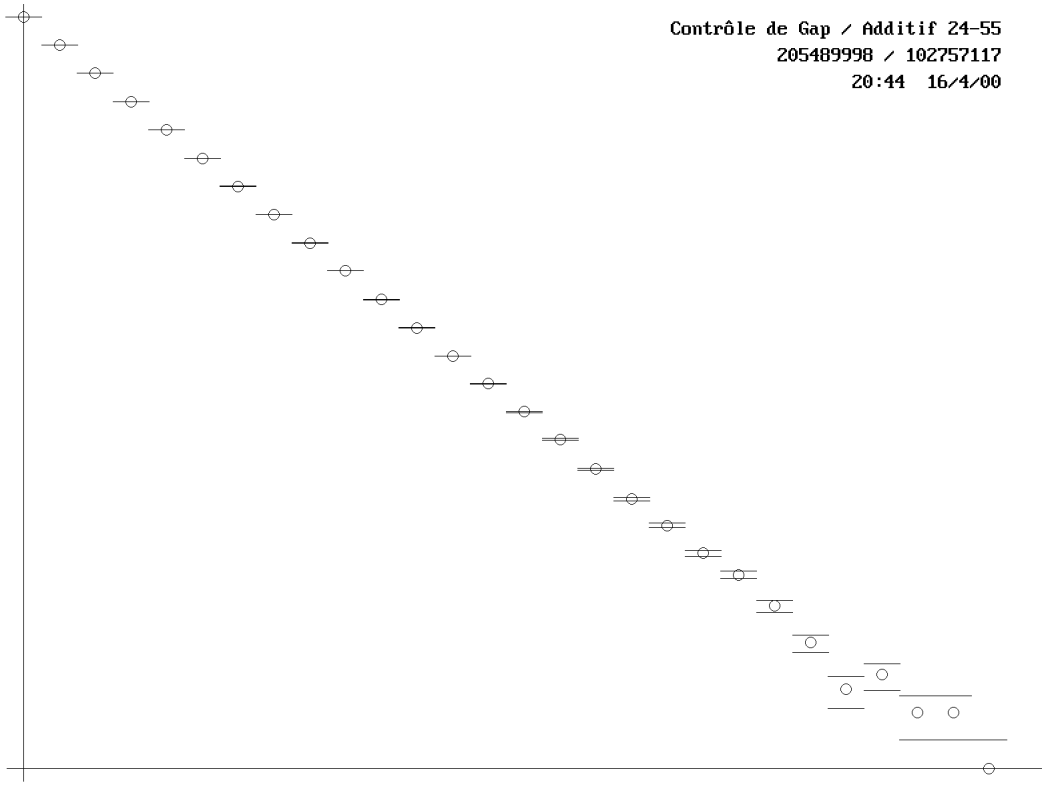

Kolmogorov Smirnov / PolynômeX 4
216625 / 1000 / 2436
15:50 21/04/00

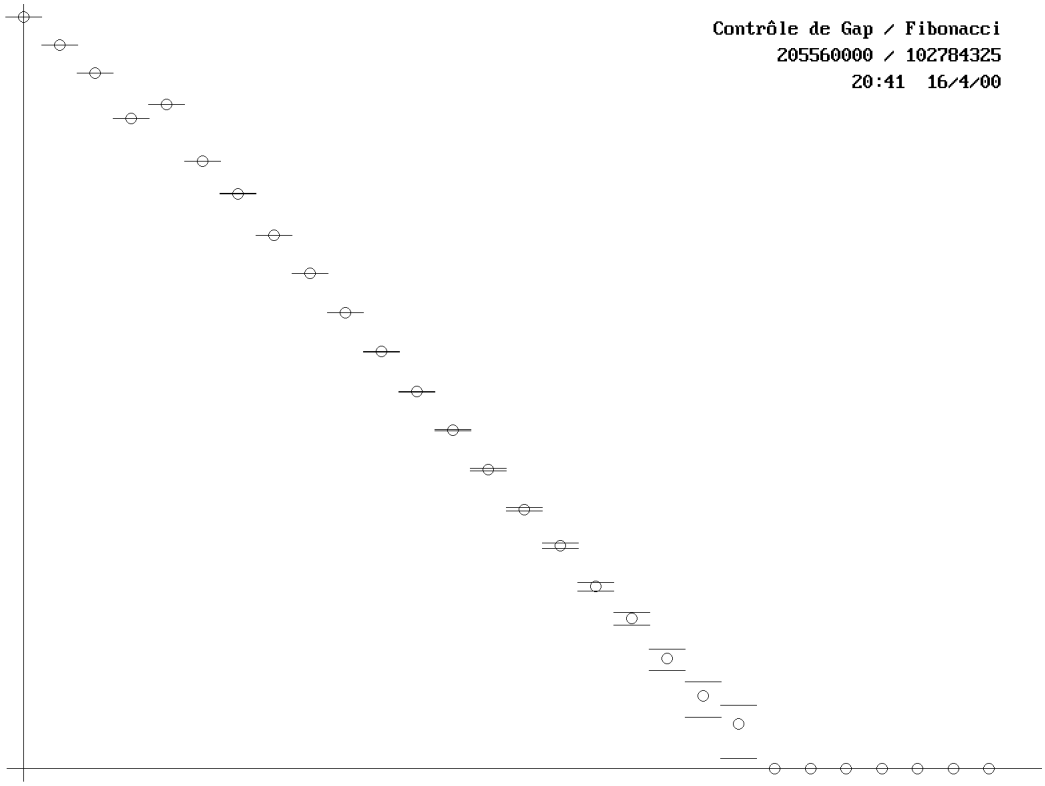

Kolmogorov Smirnov / Turbo Pascal
1845252 / 1000 / 22573
16:39 16/4/00

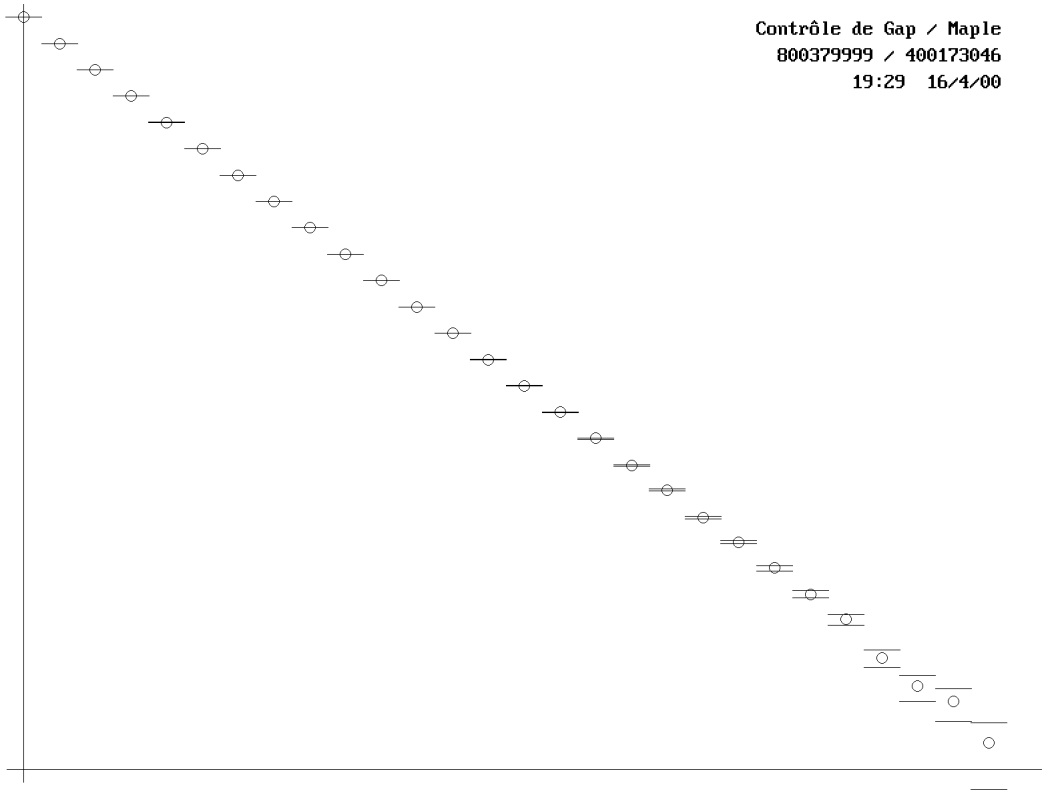
- Test d'intervalle (Gap test, D. Knuth, tome 2, page 62)

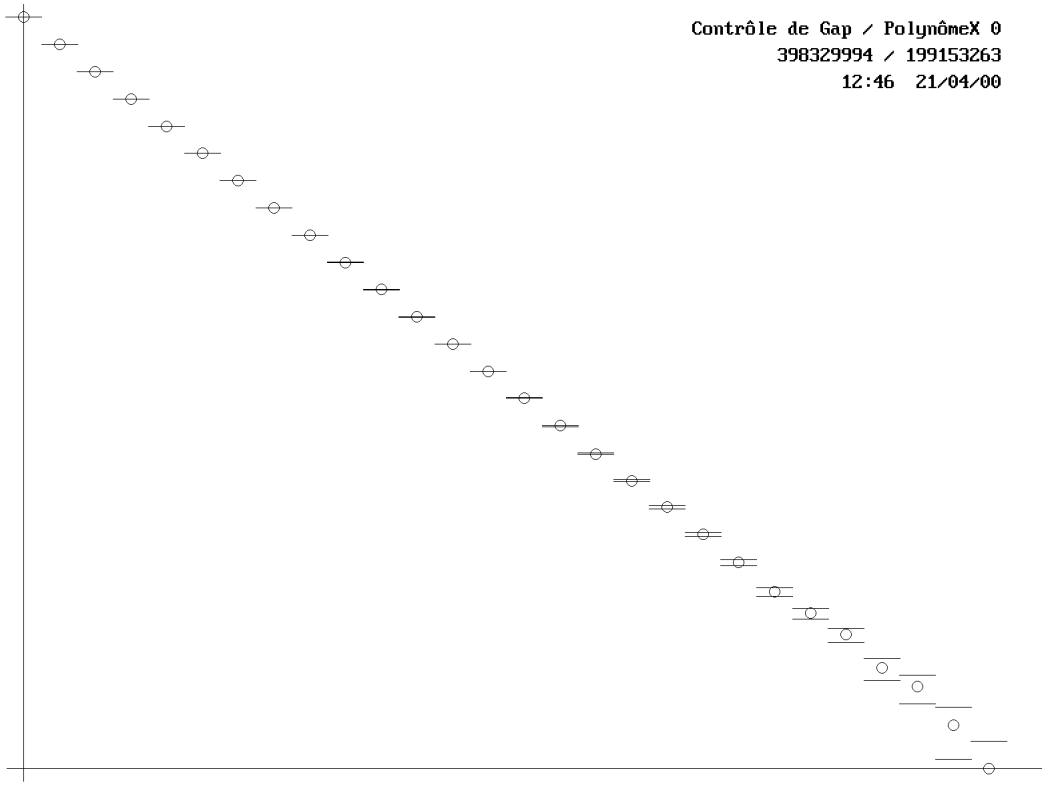
Ce test vérifie si les données sont correctement réparties dans un intervalle donné. On compte combien de fois consécutives le nombre aléatoire est situé dans l'intervalle [α, β], et on vérifie si la loi de probabilité est satisfaite.

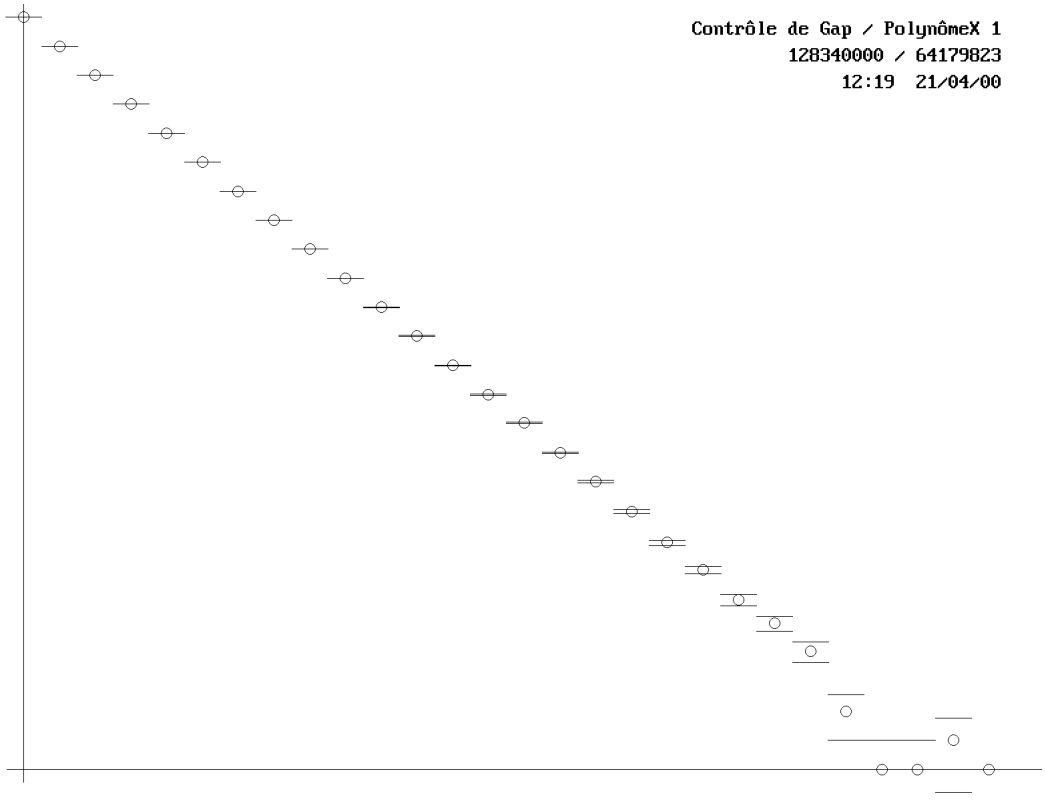
J'ai choisit $\alpha = 0$ et $\beta = 0.5$. La probabilité d'observer p événements consécutifs doit être alors égale à 2^{-p}. Les paramètres indiqués sont respectivement le nom du test, la nature du générateur, le nombre d'essais effectués, la hauteur du pic, l'heure et la date de réalisation du test.

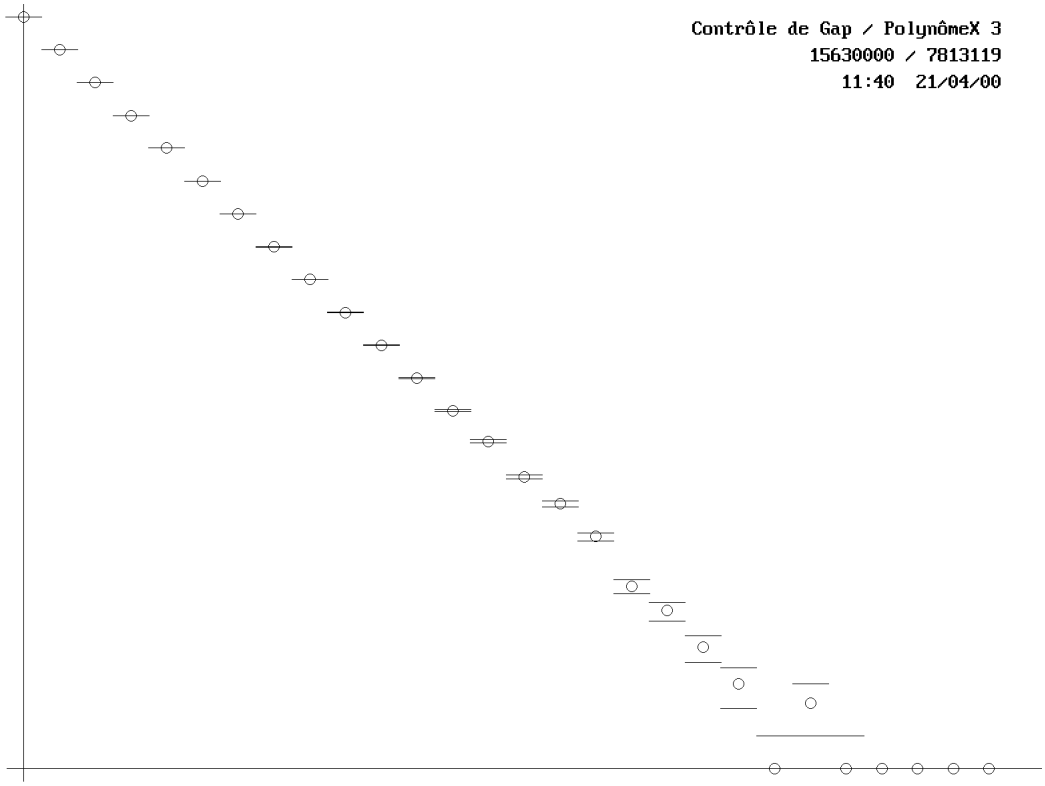


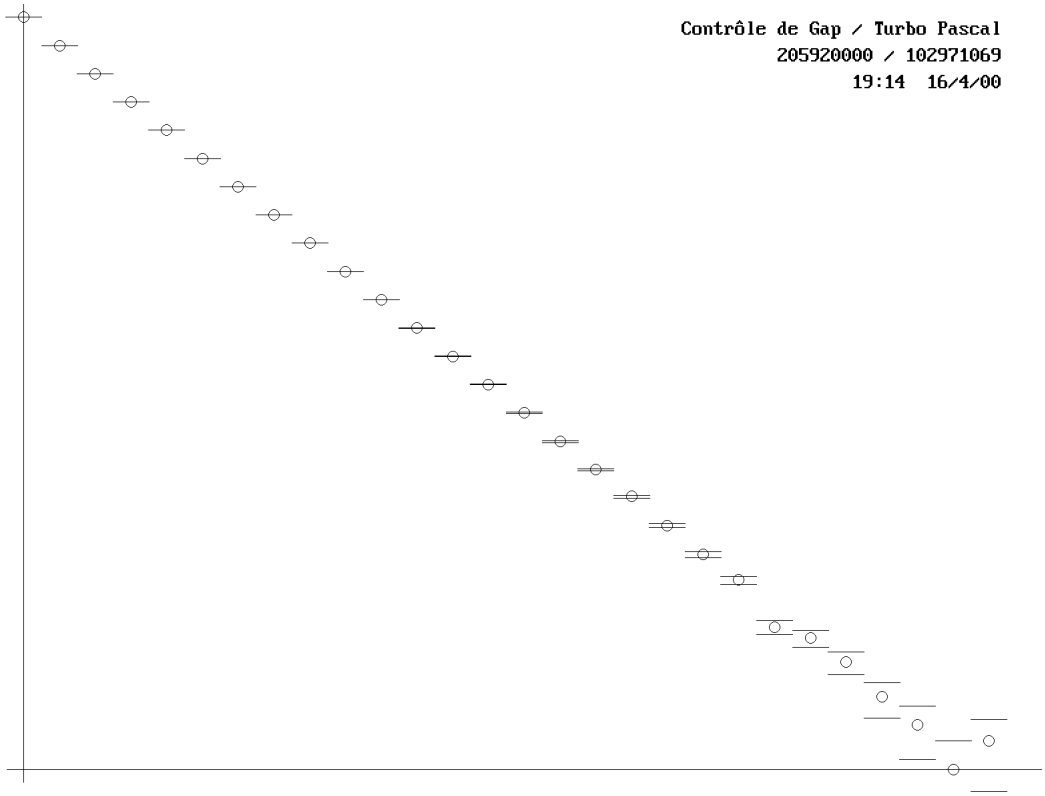

Contrôle de Gap / Fibonacci
205560000 / 102784325
20:41 16/4/00


Contrôle de Gap / Maple
800379999 / 400173046
19:29 16/4/00


Contrôle de Gap / PolynômeX 0
398329994 / 199153263
12:46 21/04/00


Contrôle de Gap / PolynômeX 1
128340000 / 64179823
12:19 21/04/00


Contrôle de Gap / PolynômeX 3
15630000 / 7813119
11:40 21/04/00


Contrôle de Gap / Turbo Pascal
205920000 / 102971069
19:14 16/4/00

- Test de corrélation série (D. Knuth, tome 2, page 72)

Après avoir enregistré N nombres aléatoires consécutifs $<U_k>$, on calcule le coefficient de corrélation entre ces nombres, et ces mêmes nombres décalés de d, soit $<V_k>$, où $V_k = U_{(k+d) \bmod N}$. Le coefficient de corrélation a la valeur :

$$C = \frac{n\sum_j (U_j V_j) - \left(\sum_j U_j\right)\left(\sum_j V_j\right)}{\sqrt{\left(n\sum_j U_j^2 - \left(\sum_j U_j\right)^2\right)\left(n\sum_j V_j^2 - \left(\sum_j V_j\right)^2\right)}}$$

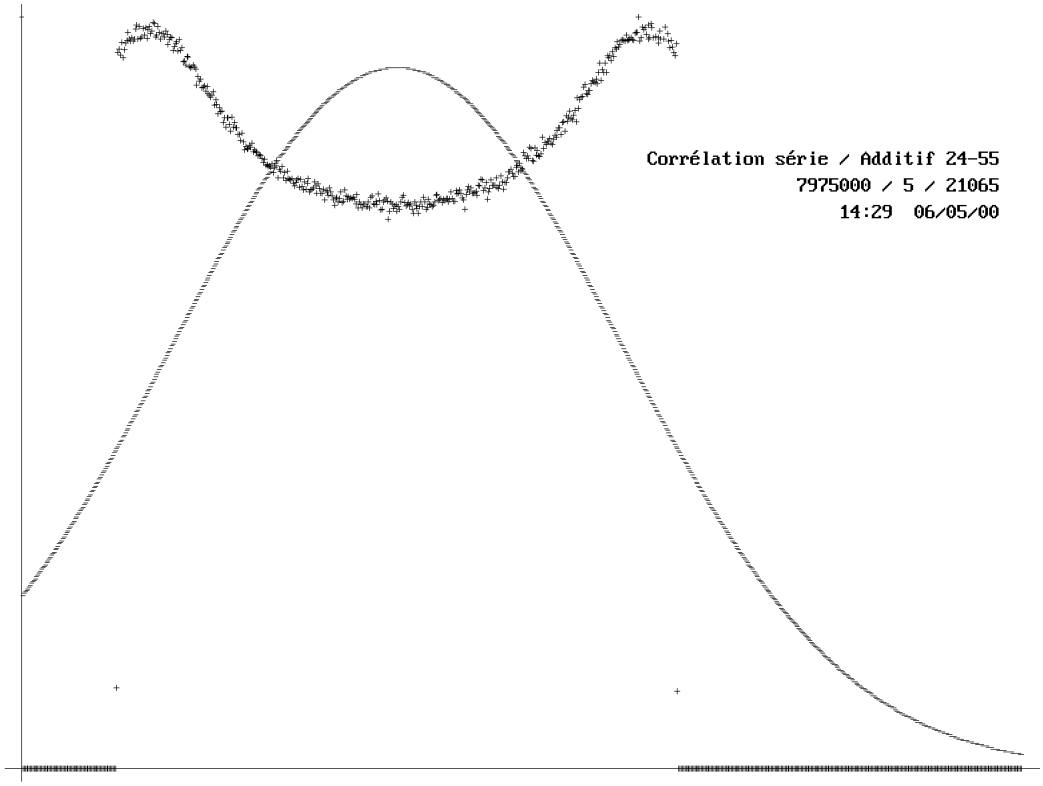
Il est compris entre -1 et 1. Il est proche de zéro quand les grandeurs U_k et V_k sont indépendantes les unes des autres.

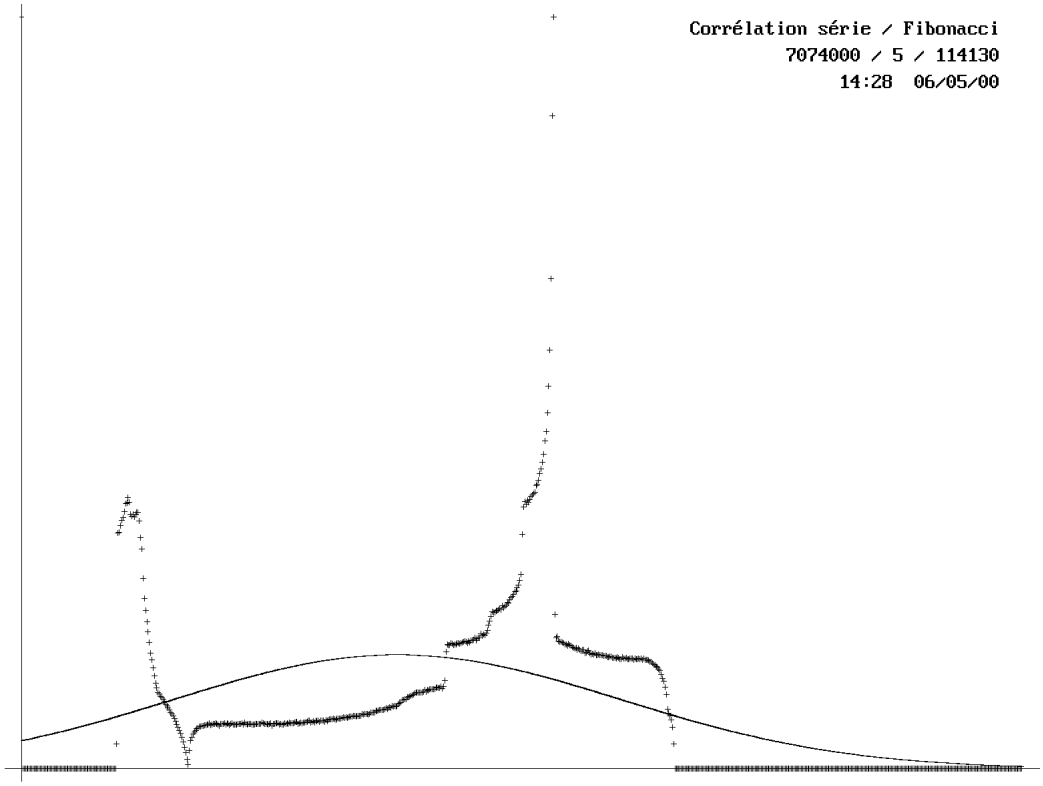
Comme ce n'est pas tout à fait le cas, on obtient une courbe de distribution du coefficient de corrélation, qui est proche d'une courbe gaussienne (pour des valeurs de N assez grandes), centrée sur la valeur μ_N, et de largeur σ_N :

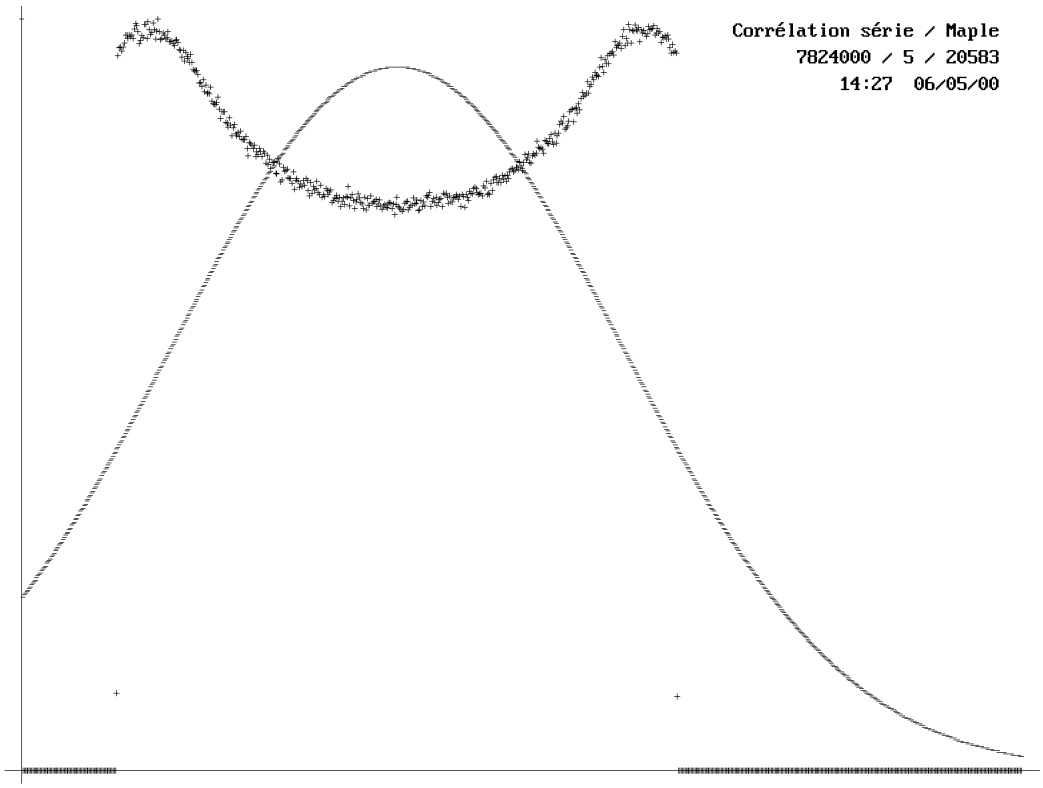
$$\mu_N = \frac{-1}{N-1}, \qquad \sigma_N^2 = \frac{N^2}{(N-1)^2 (N-2)}, \qquad \text{pour } N > 9$$

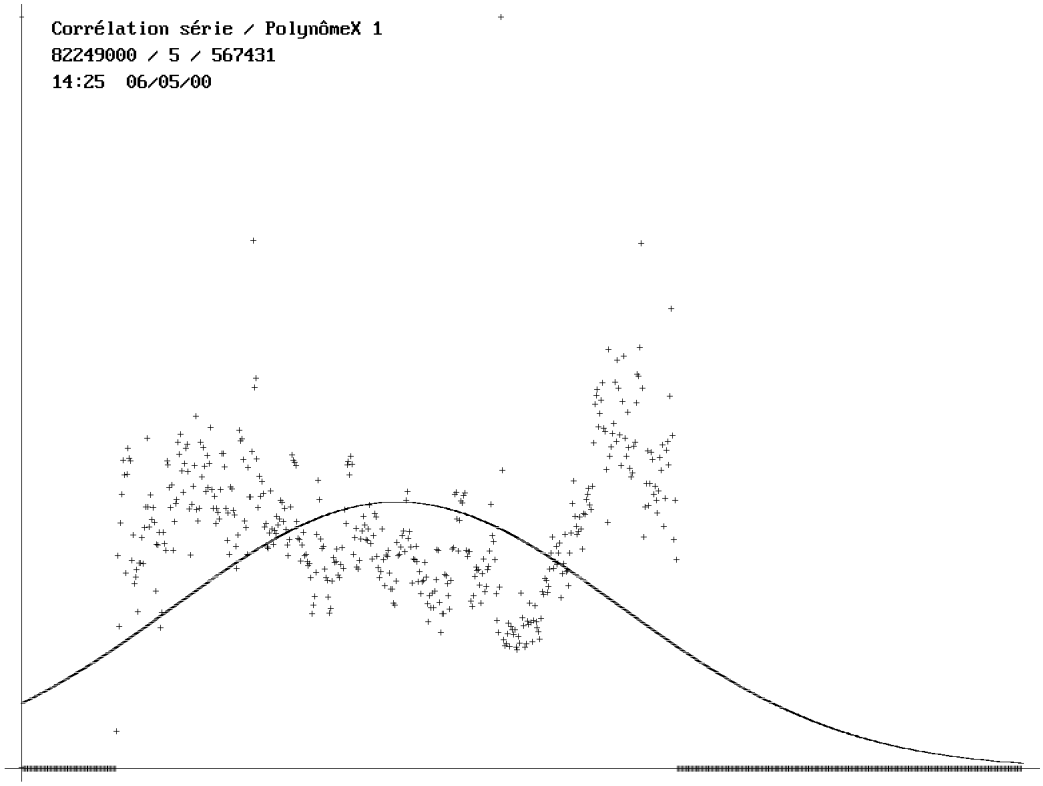
Ceci permet de contrôler la validité des résultats.

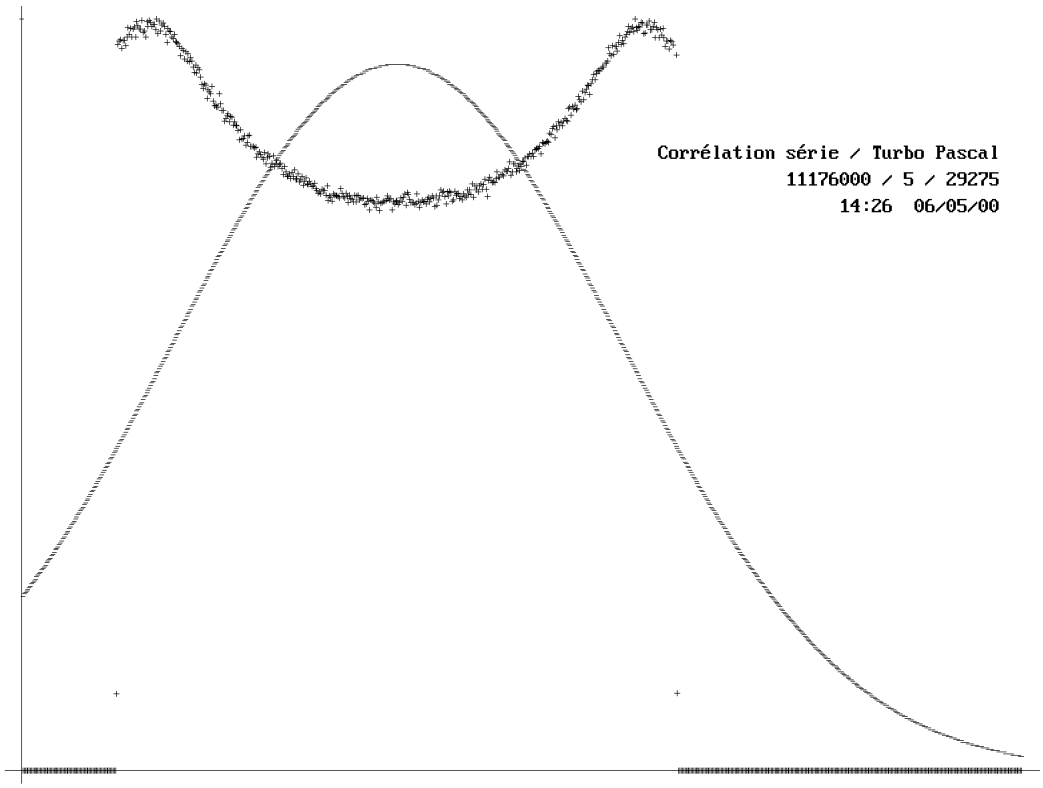
Tous les tests ont été effectués avec un décalage $d = 1$. Les paramètres indiqués sont respectivement le nom du test, la nature du générateur, le nombre d'essais effectués, la valeur de N, la hauteur du pic, l'heure et la date de réalisation du test.

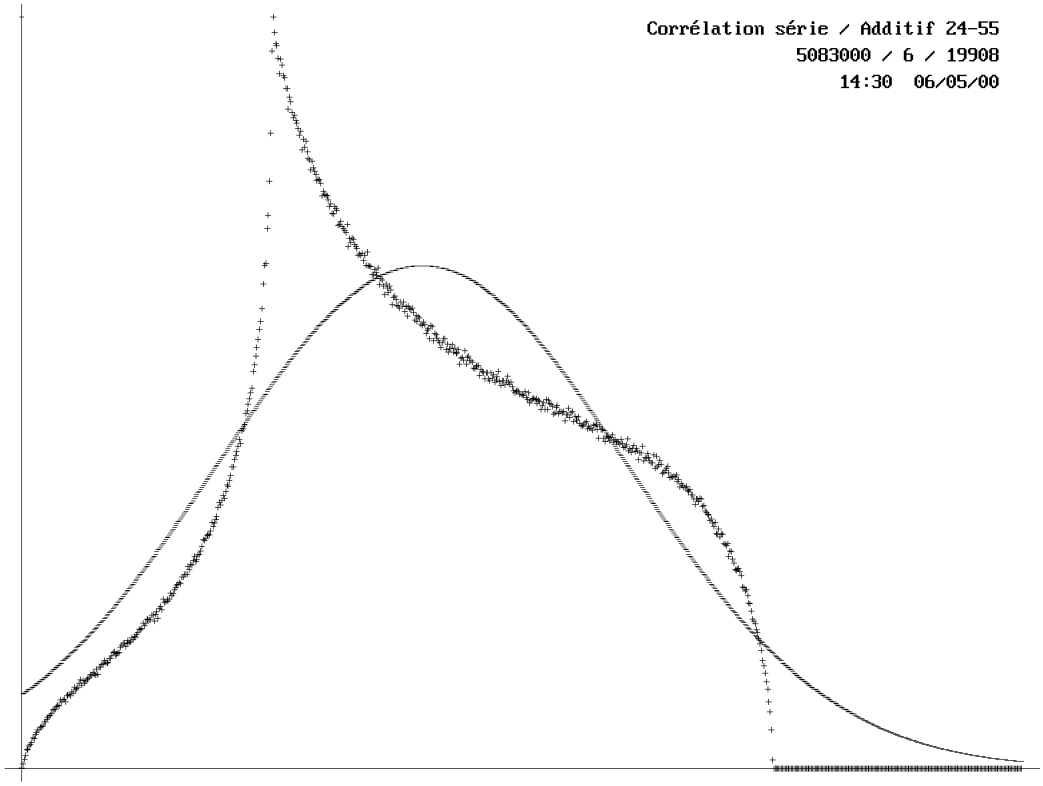


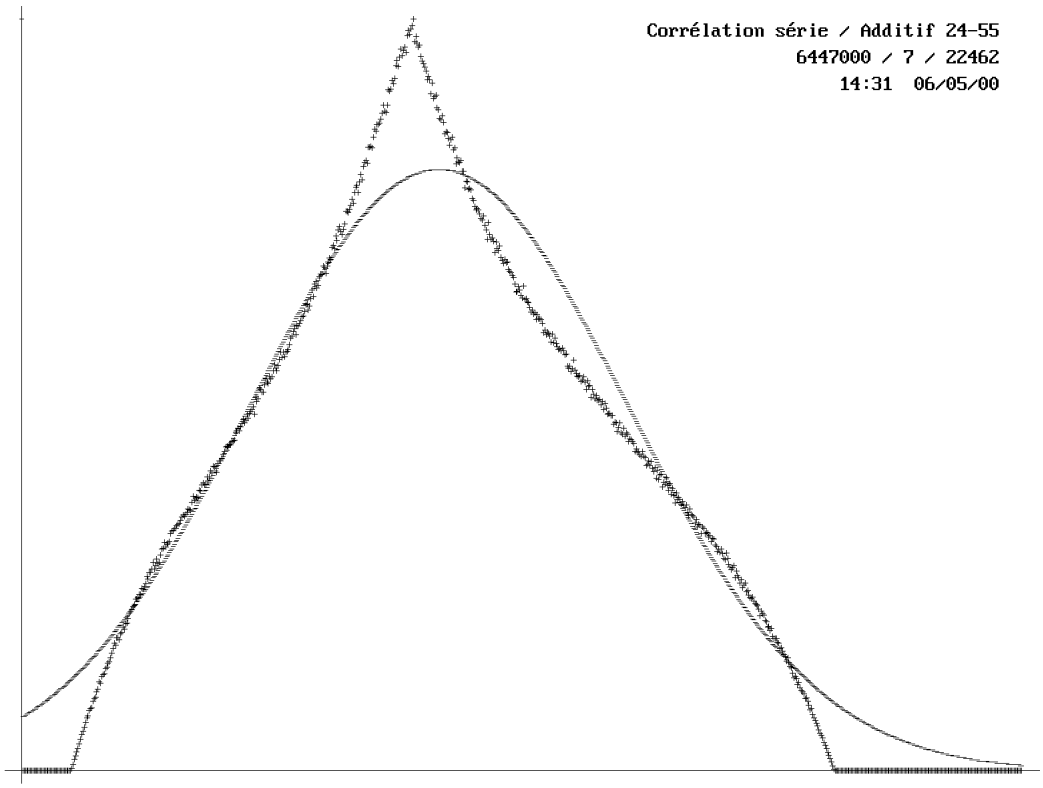

Corrélation série / Fibonacci
7074000 / 5 / 114130
14:28 06/05/00

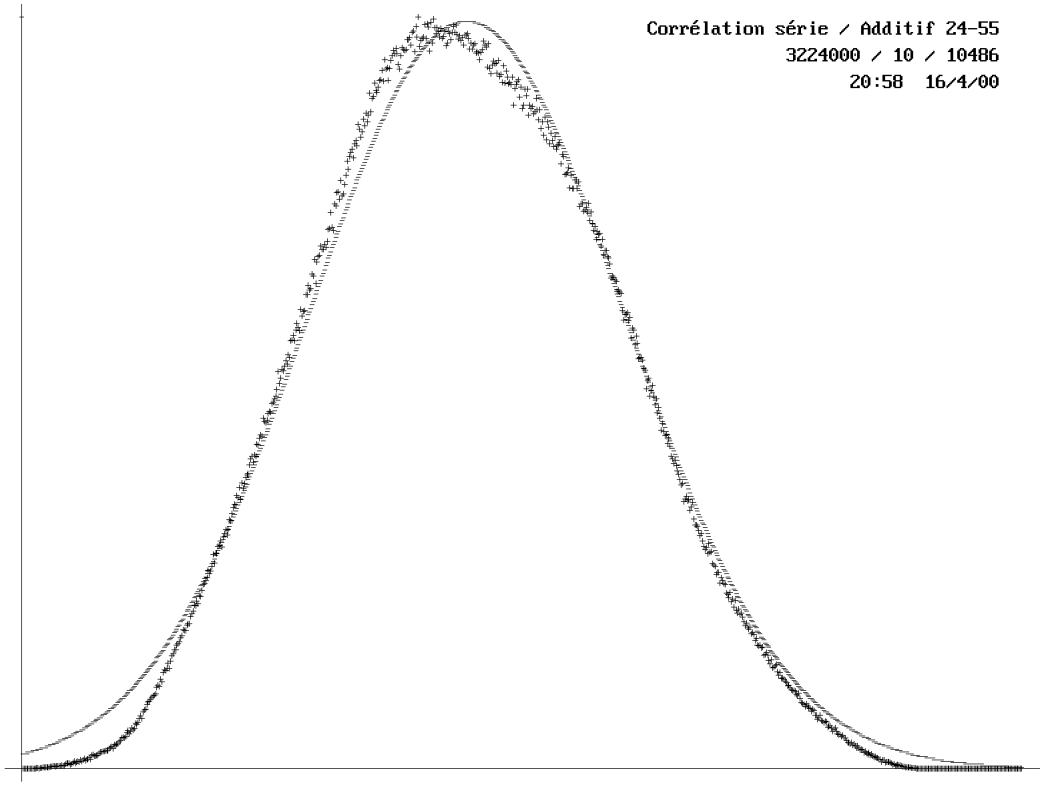

Corrélation série / Maple
7824000 / 5 / 20583
14:27 06/05/00

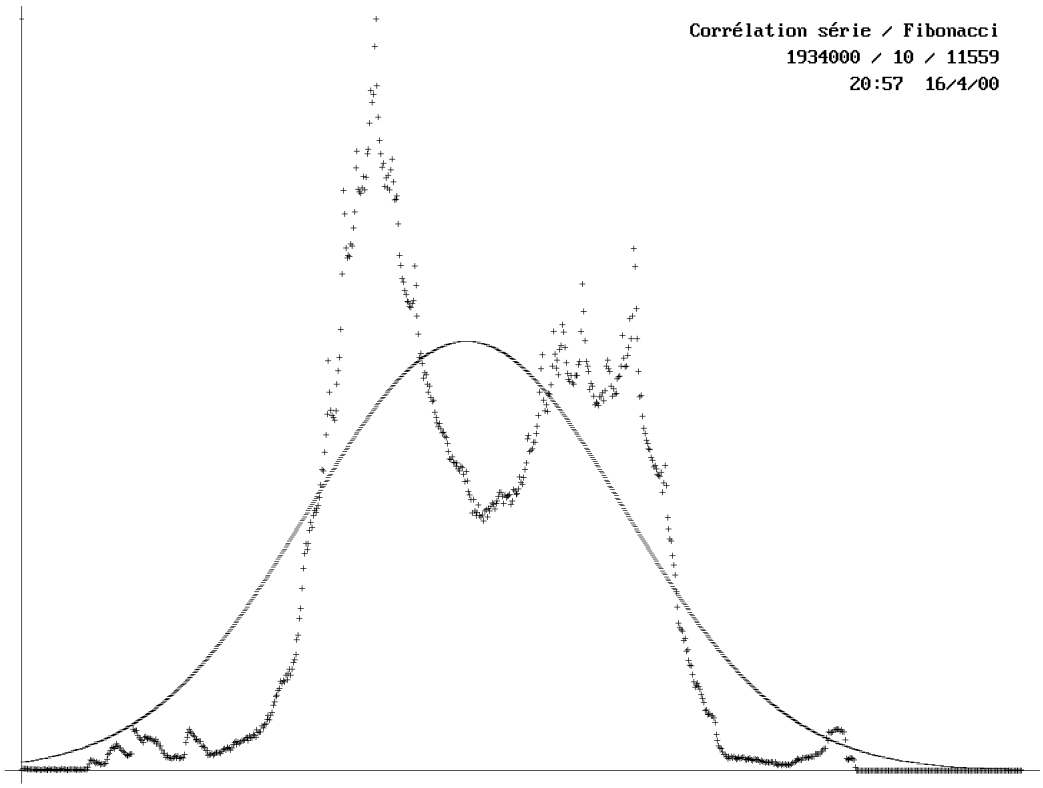

Corrélation série / PolynômeX 1
82249000 / 5 / 567431
14:25 06/05/00

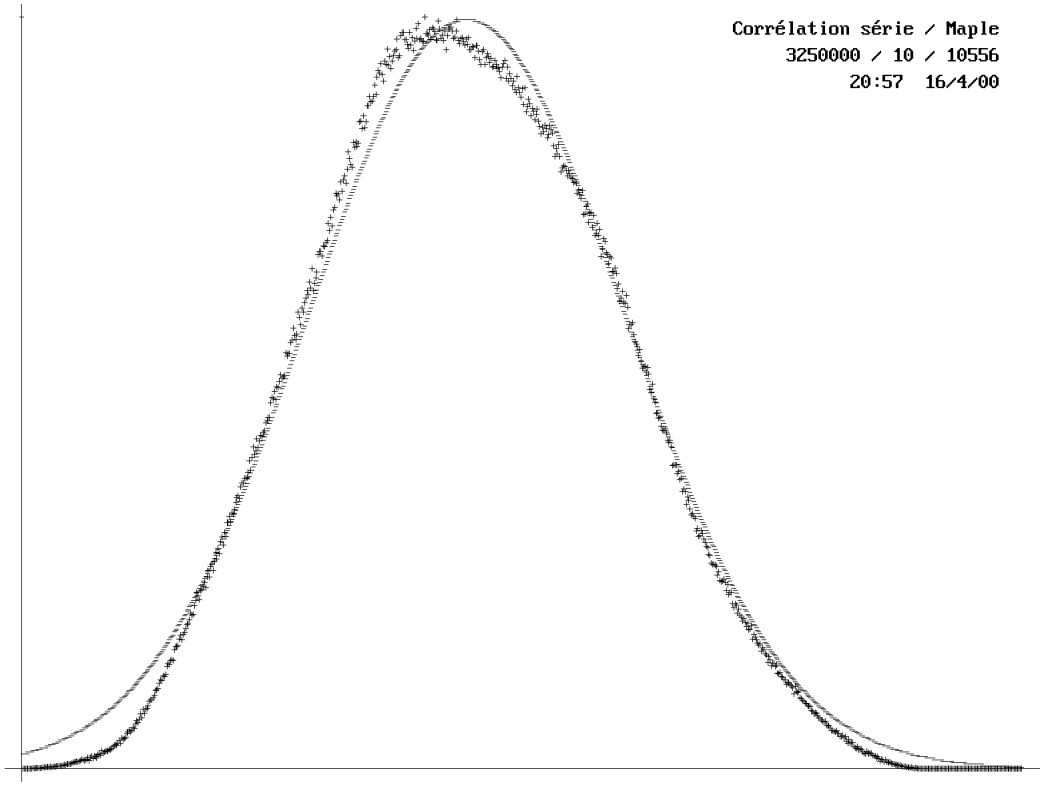

Corrélation série / Turbo Pascal
11176000 / 5 / 29275
14:26 06/05/00

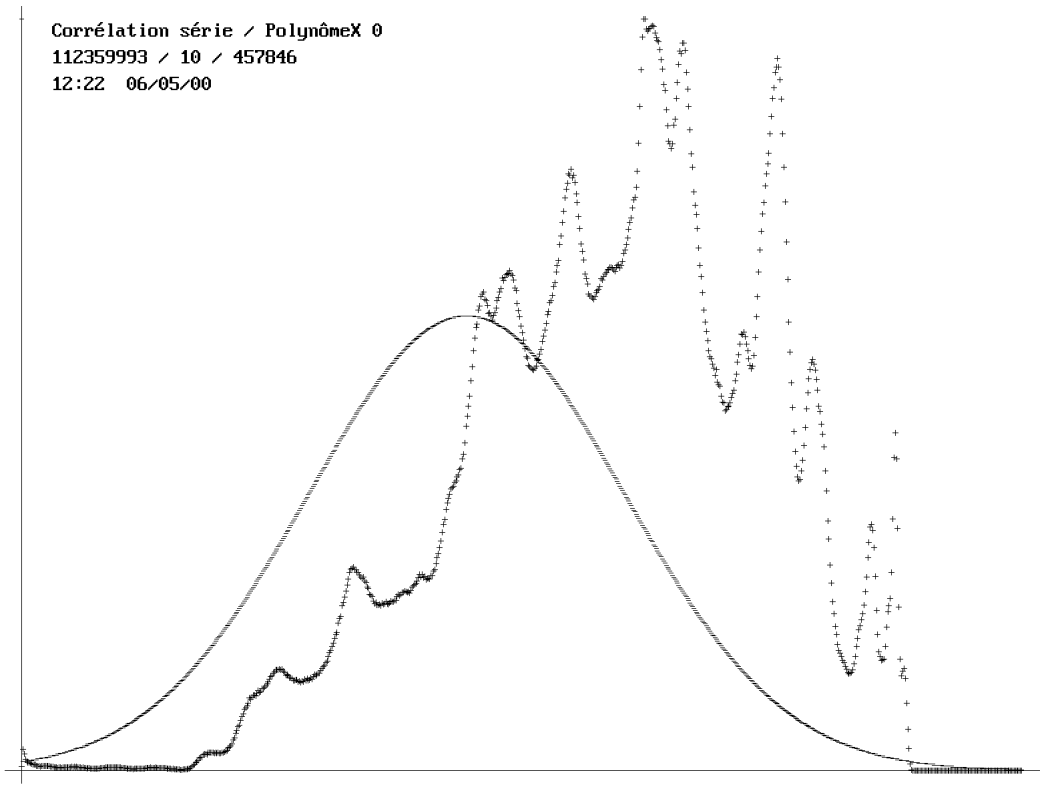

Corrélation série / Additif 24-55
5083000 / 6 / 19908
14:30 06/05/00

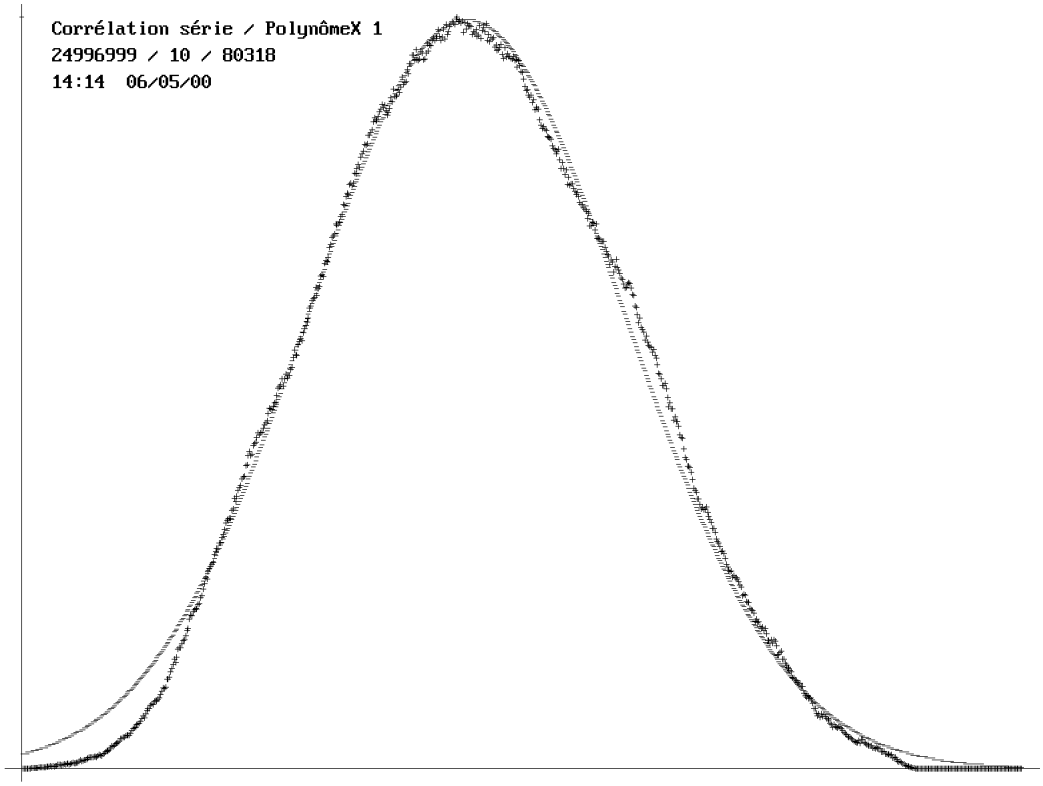

Corrélation série / Additif 24-55
6447000 / 7 / 22462
14:31 06/05/00

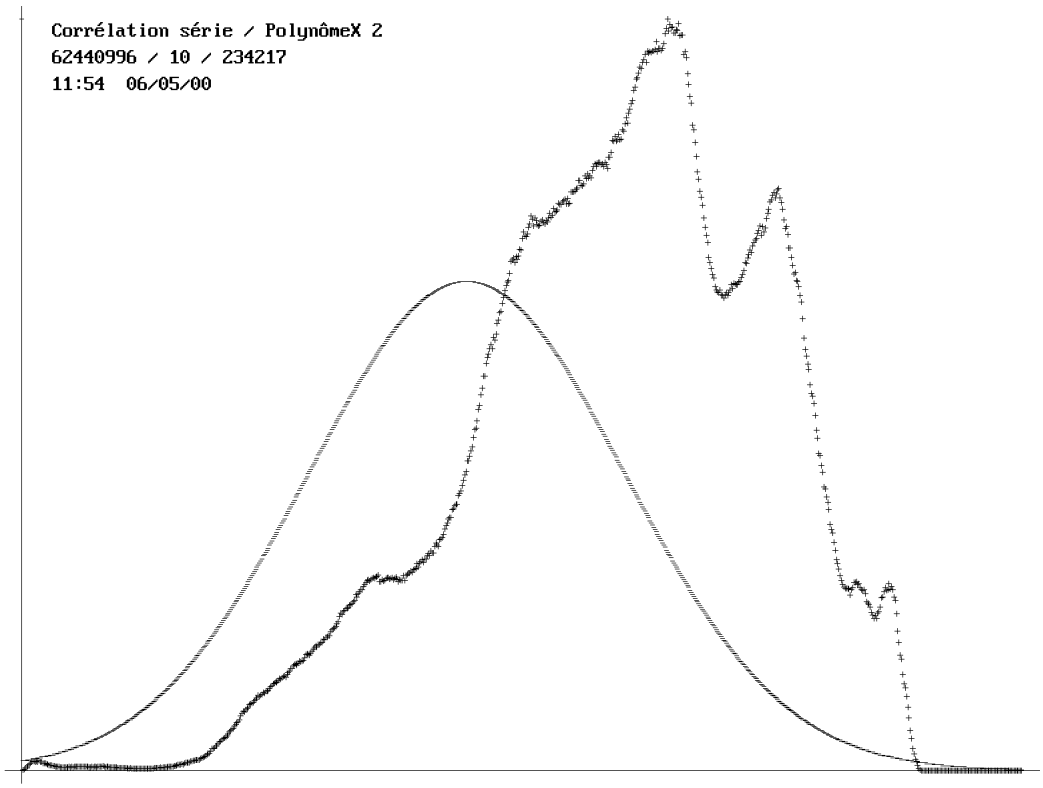

Corrélation série / Additif 24-55
3224000 / 10 / 10486
20:58 16/4/00

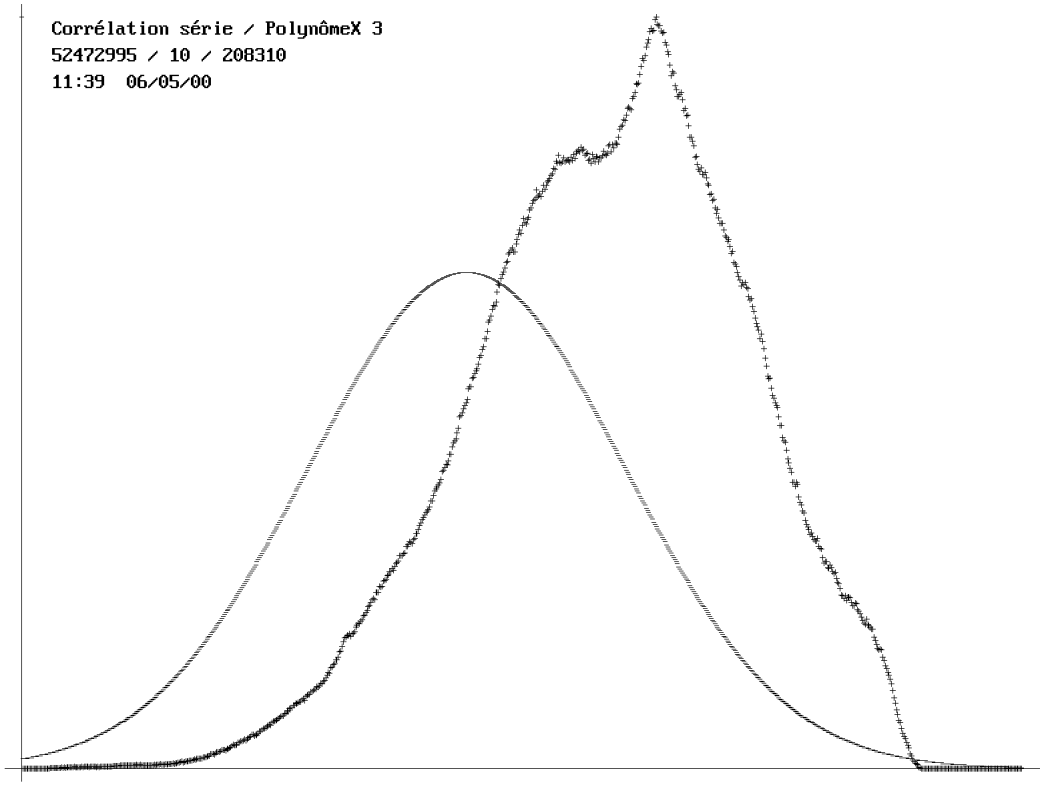

Corrélation série / Fibonacci
1934000 / 10 / 11559
20:57 16/4/00

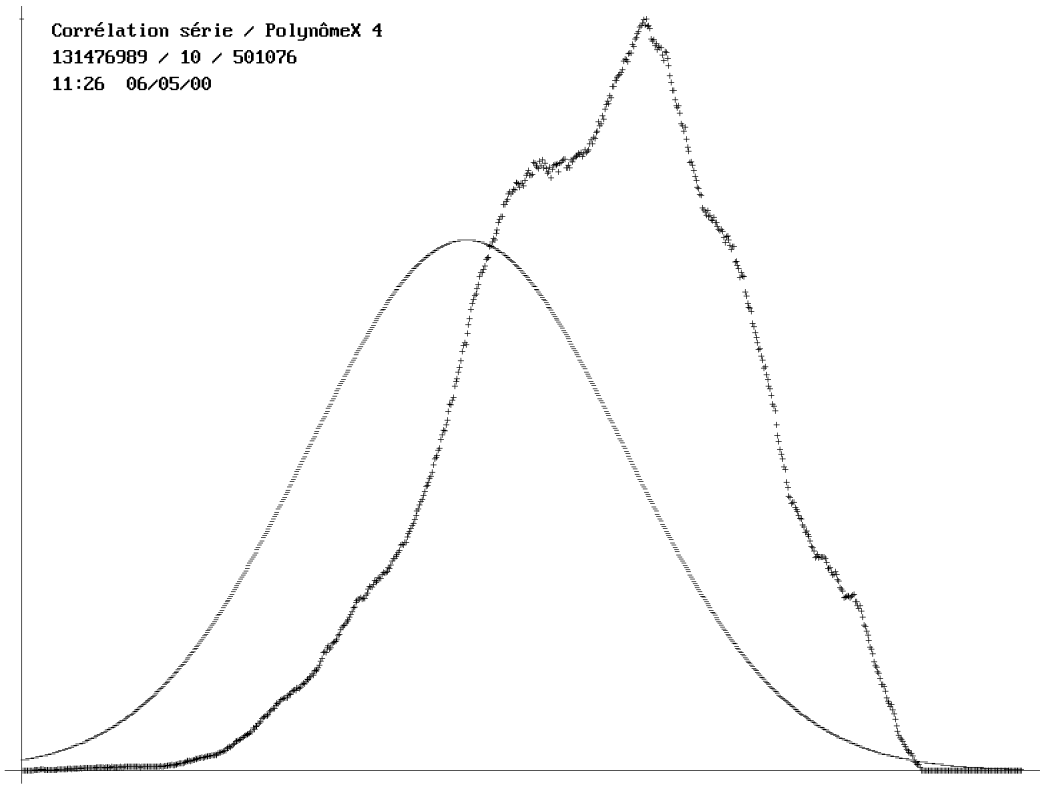

Corrélation série / Maple
3250000 / 10 / 10556
20:57 16/4/00

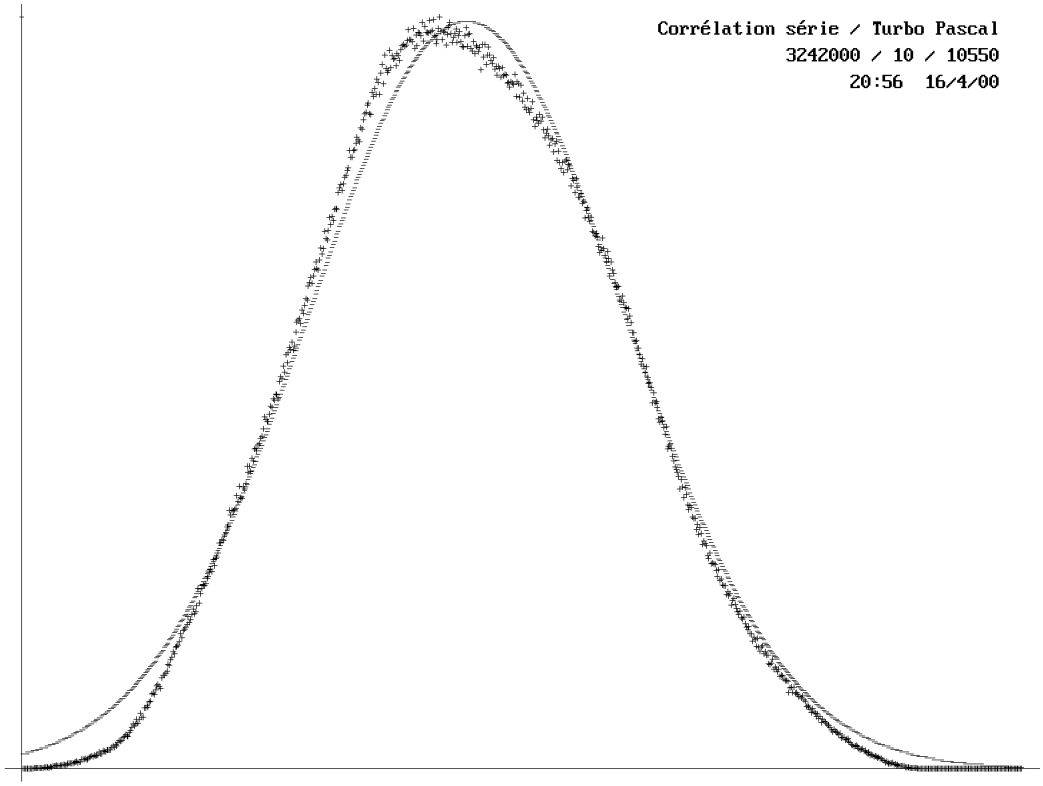

Corrélation série / PolynômeX 0
112359993 / 10 / 457846
12:22 06/05/00

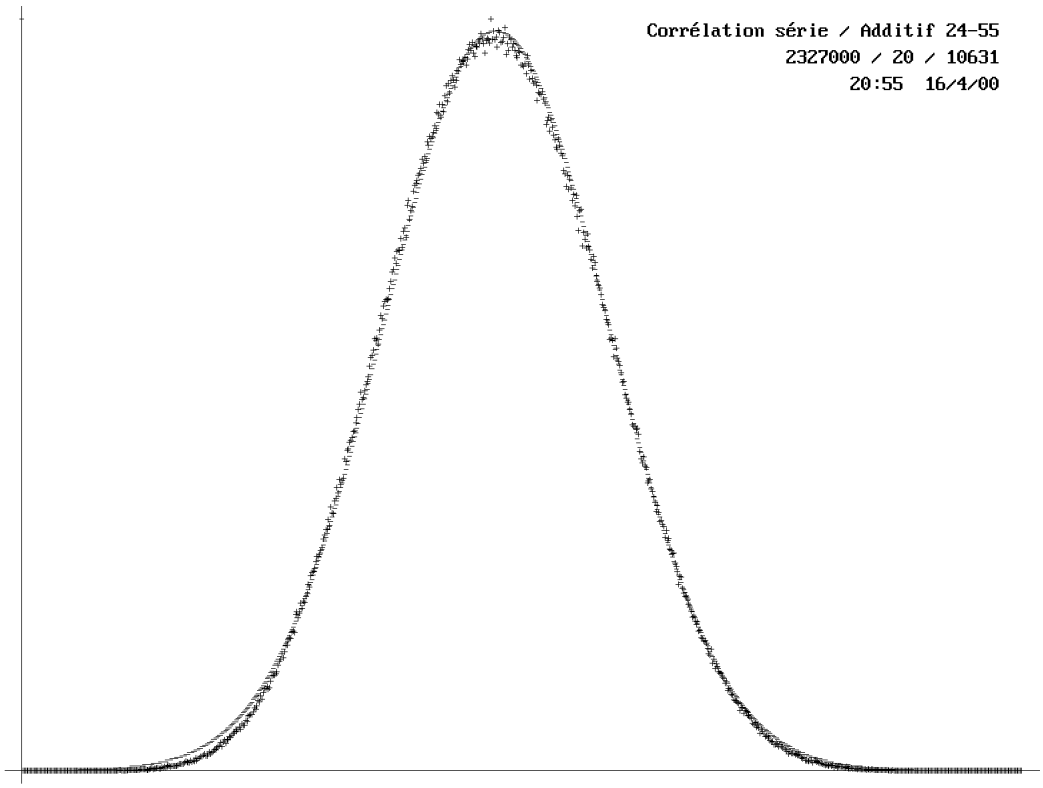

Corrélation série / PolynômeX 1
24996999 / 10 / 80318
14:14 06/05/00

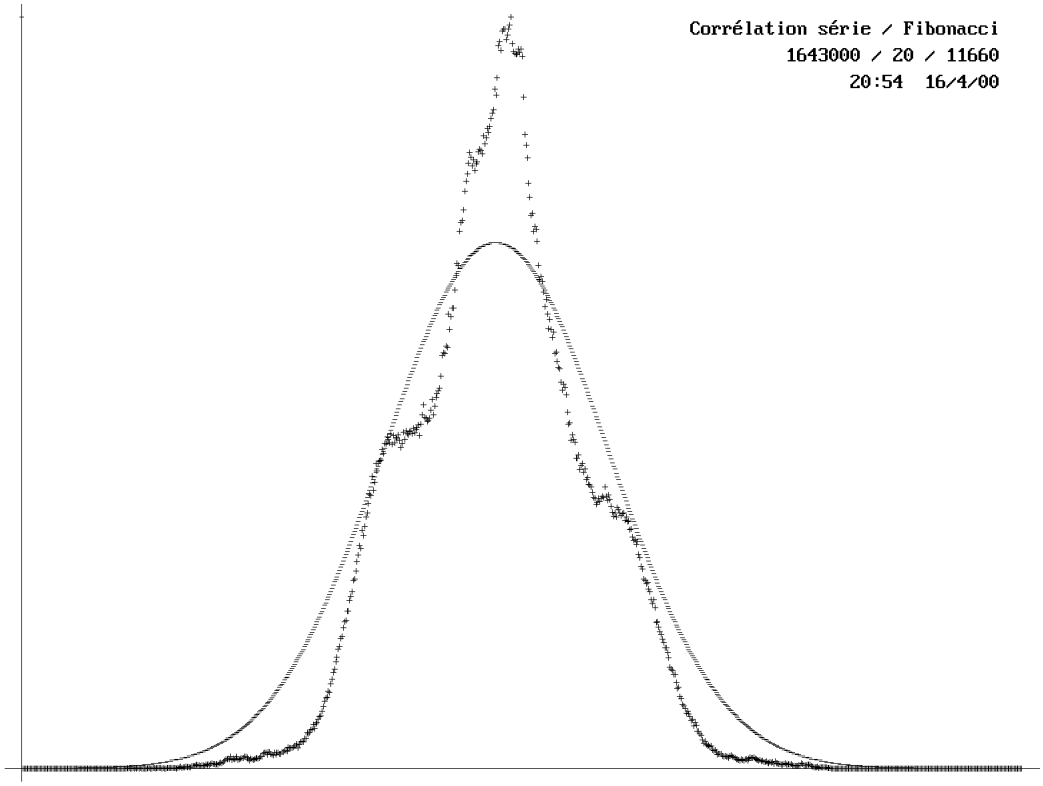

Corrélation série / PolynômeX 2
62440996 / 10 / 234217
11:54 06/05/00

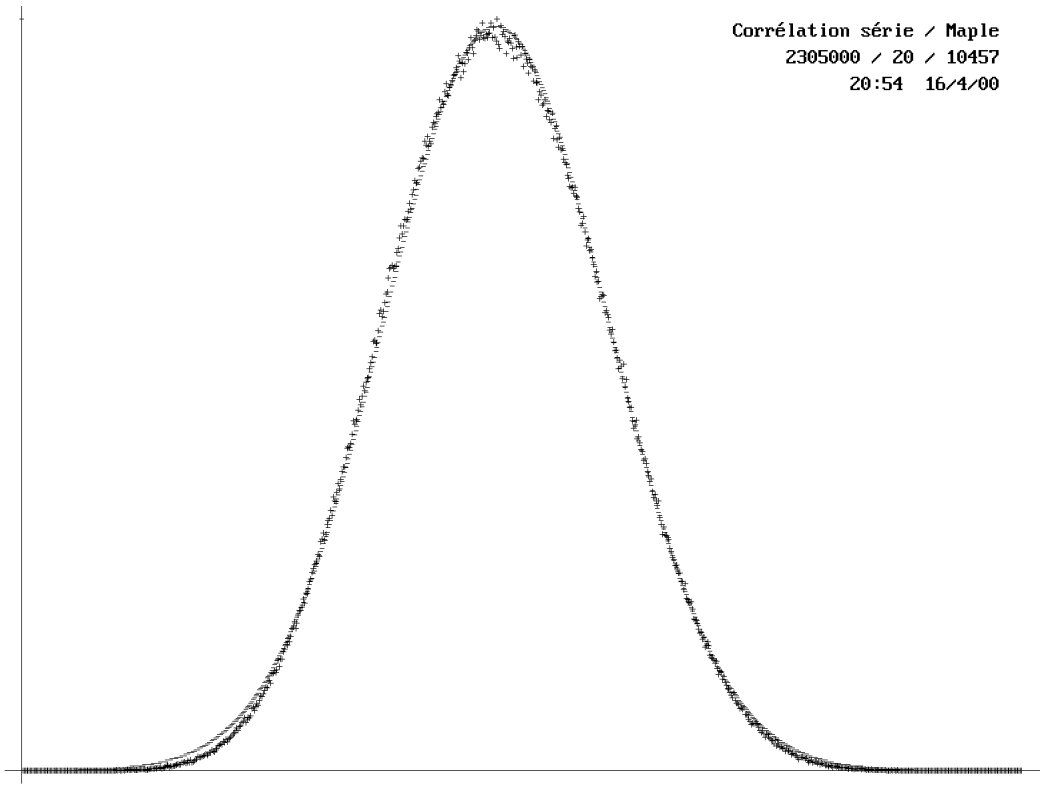

Corrélation série / PolynômeX 3
52472995 / 10 / 208310
11:39 06/05/00

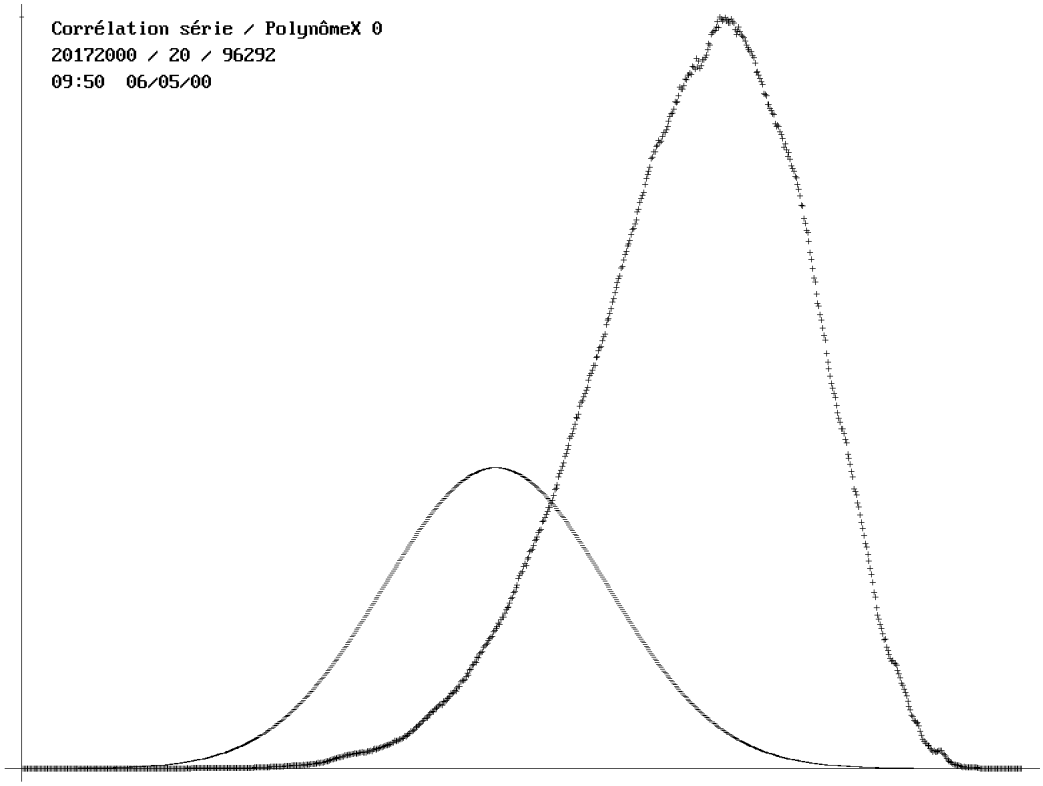

Corrélation série / PolynômeX 4
131476989 / 10 / 501076
11:26 06/05/00

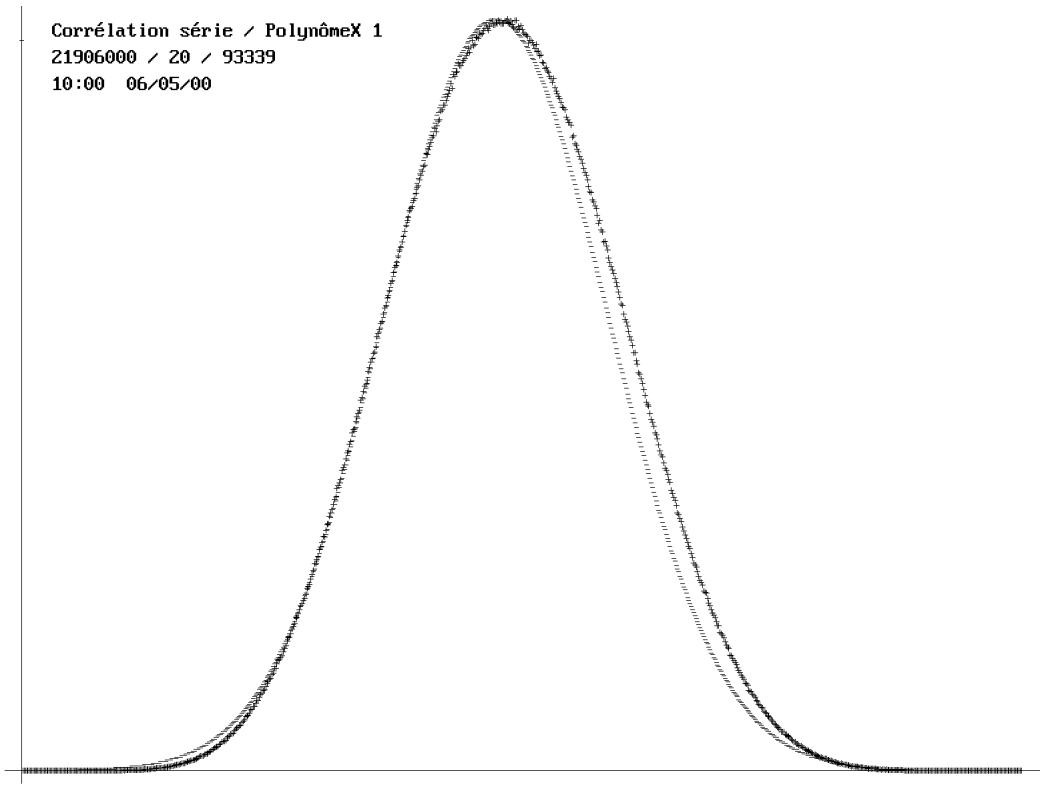

Corrélation série / Turbo Pascal
3242000 / 10 / 10550
20:56 16/4/00

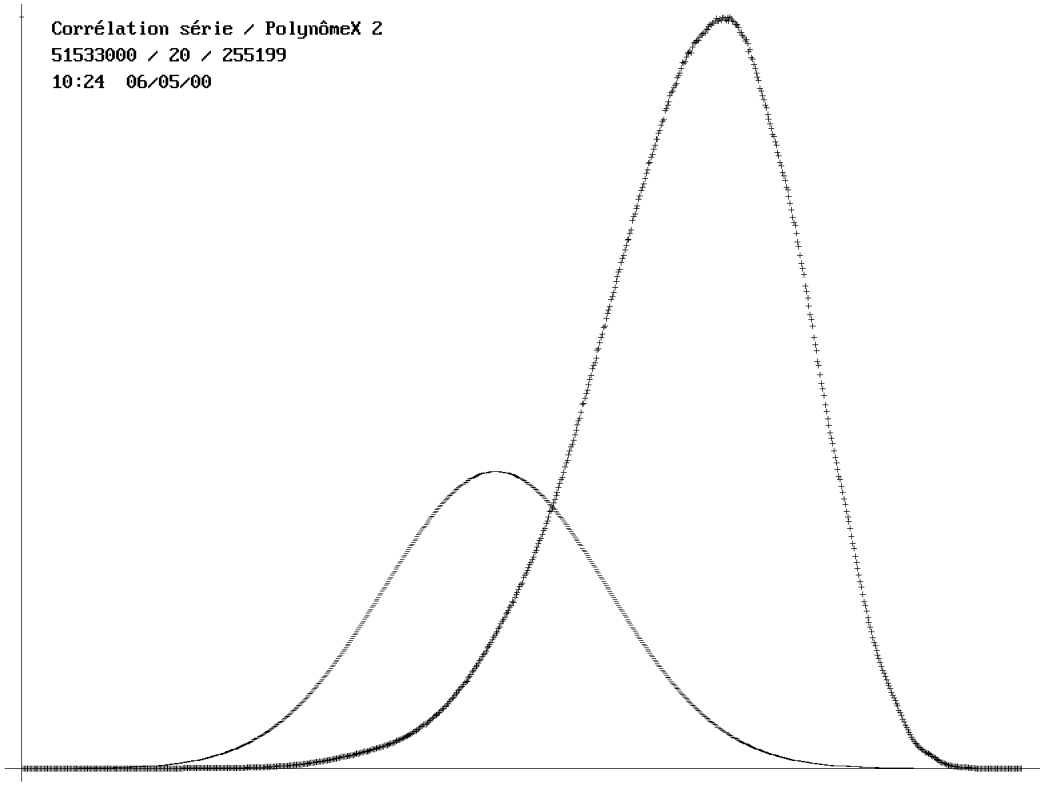

Corrélation série / Additif 24-55
2327000 / 20 / 10631
20:55 16/4/00

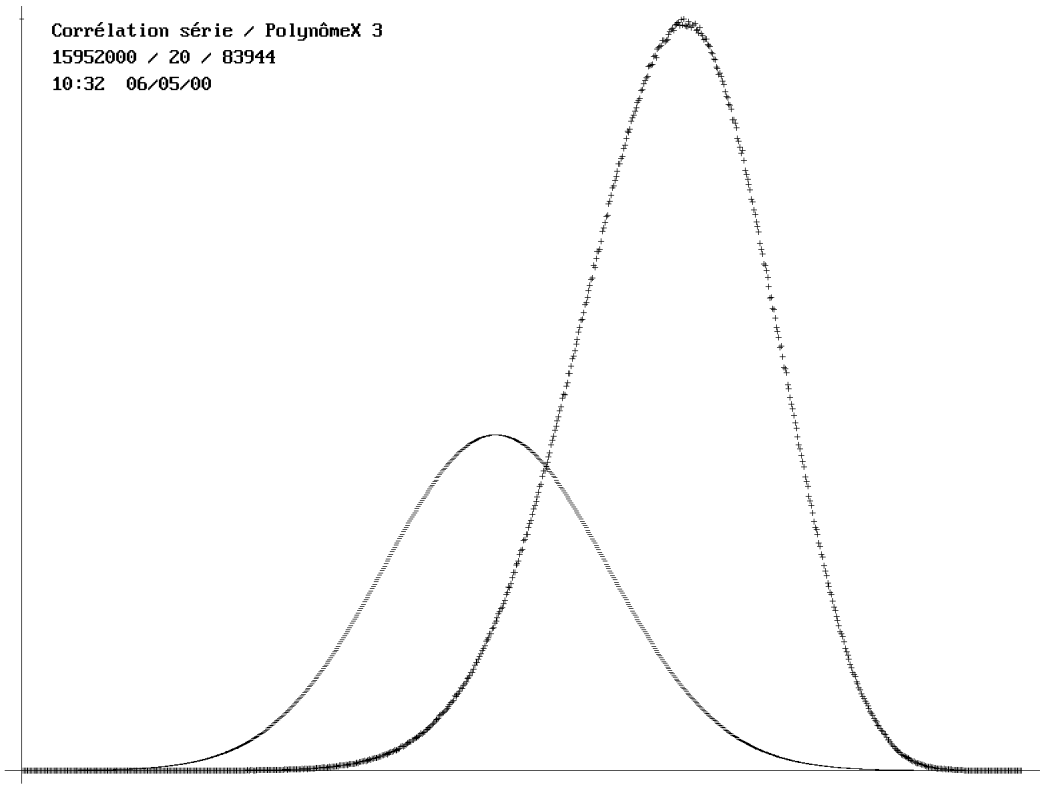

Corrélation série / Fibonacci
1643000 / 20 / 11660
20:54 16/4/00

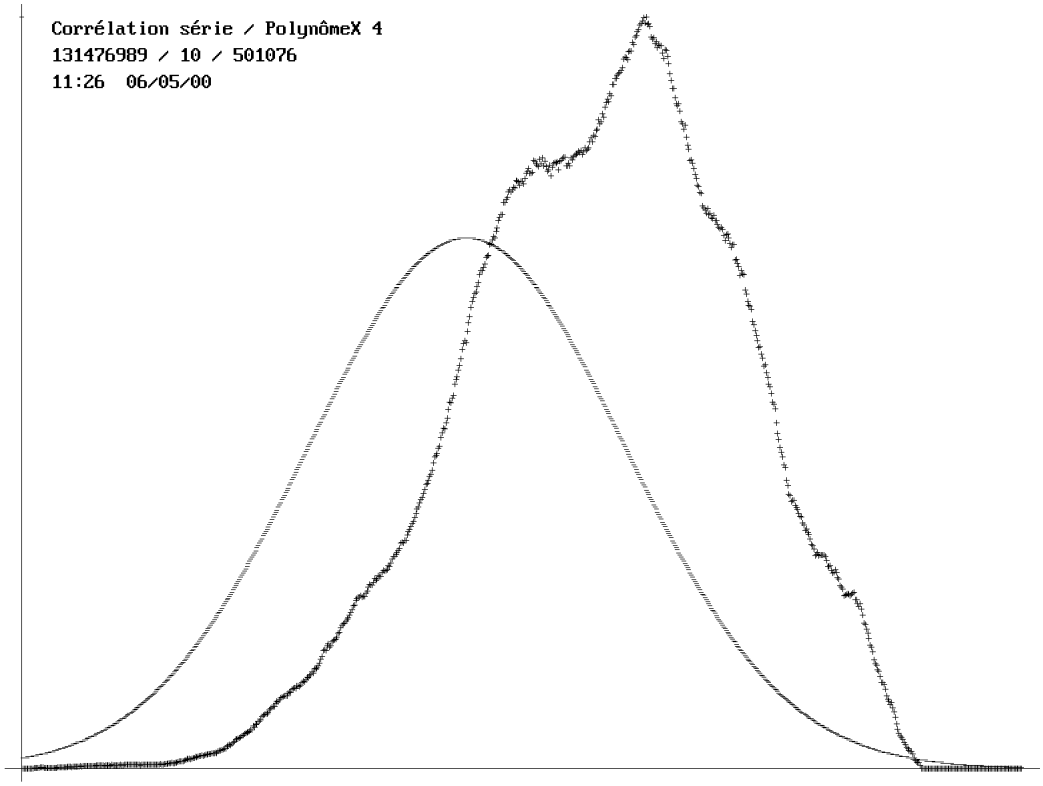

Corrélation série / Maple
2305000 / 20 / 10457
20:54 16/4/00

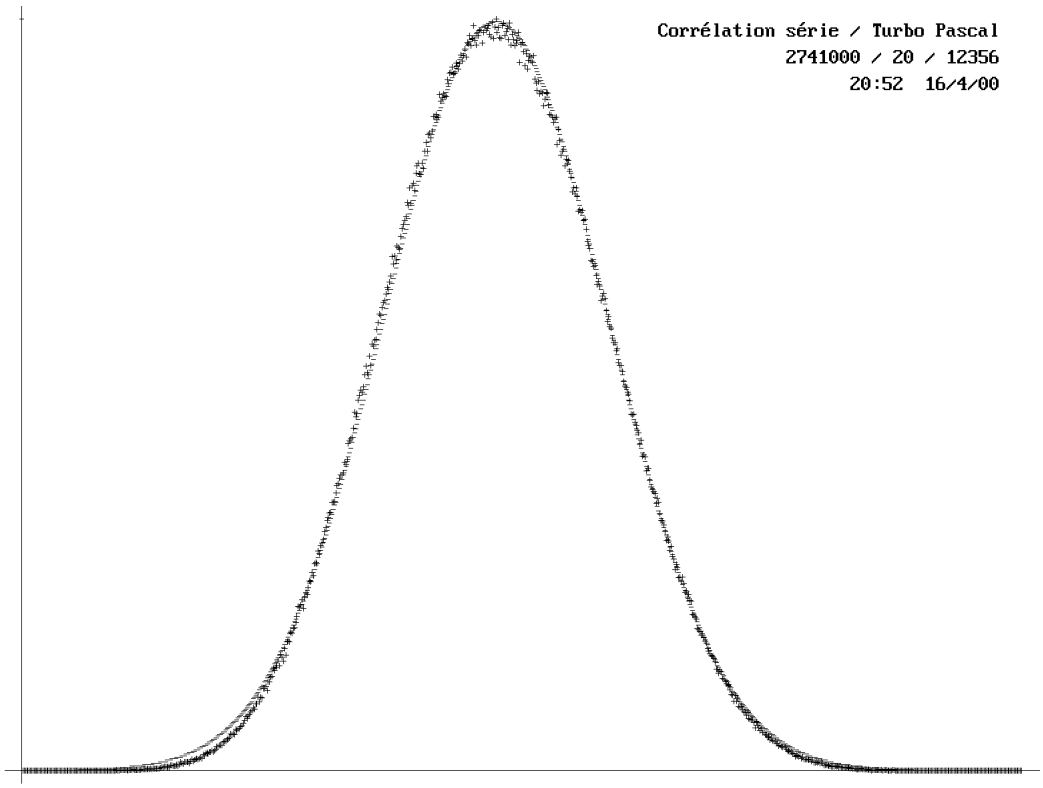

Corrélation série / PolynômeX 0
20172000 / 20 / 96292
09:50 06/05/00


Corrélation série / PolynômeX 1
21906000 / 20 / 93339
10:00 06/05/00


Corrélation série / PolynômeX 2
51533000 / 20 / 255199
10:24 06/05/00


Corrélation série / PolynômeX 3
15952000 / 20 / 83944
10:32 06/05/00


Corrélation série / PolynômeX 4
131476989 / 10 / 501076
11:26 06/05/00


Corrélation série / Turbo Pascal
2741000 / 20 / 12356
20:52 16/4/00

- Test spectral (D. Knuth, tome 2, pages 93-118)

C'est un test théorique (par rapport aux précédents, qui sont plutôt expérimentaux).

On considère un générateur à congruence linéaire tel que $X_{n+1} = (aX_n + c) \bmod m$, pour lequel la période est m ou pour lequel $c = 0$, m est premier et la période égale à $m - 1$.

On considère l'ensemble des m (ou $m - 1$) points de l'espace à dimension t, dont les coordonnées sont du type $(X_p, X_{p+1}, ... X_{p+t-1})$. On calcule la distance minimale entre ces points. Le générateur est d'autant meilleur que cette distance minimale est plus grande.

Ce test est difficile à mettre en oeuvre. Il permet de déterminer les meilleures valeurs des constants utilisées dans les générateurs à congruence linéaire.

VI. GÉNÉRATEUR ALÉATOIRE DE MAPLE

Pourquoi choisir le générateur à congruence linéaire de Maple ?

$$X_{n+1} = aX_n \bmod m$$

pour lequel $m = 999999999989 = 10^{12} - 11$ est premier, $a = 427419669081$.

C'est un bon générateur, qui est bien testé. Il passe très bien le test spectral, comme il est indiqué dans l'article de Maple Tech.

C'est un générateur qui permet de calculer des points intermédiaires, et donc d'arrêter les calculs, puis de les redémarrer, avec un jeu nouveau de valeur aléatoires, indépendantes des précédentes. Cette propriété n'est pas vérifiée par les autres générateurs, pour lesquels on n'a pas la garantie d'obtenir une séquence différente de l'une des séquences précédentes.

- factorisation de *a*

Une difficulté réside dans le fait que le produit $a \times m$ est de l'ordre de 10^{24}, soit environ 79 bits, supérieur à la précision de calcul du processeur. Il est nécessaire d'effectuer des calculs en multiprécision ou bien de décomposer a en produit de facteurs premiers.

Heureusement a se décompose sous la forme :

$$a = 427419669081 = 3 \times 61 \times 491 \times 4756877 = 89853 \times 4756877 = a_0 \times a_1$$

ce qui est un peu à la limite de l'acceptable puisque $4.76\ 10^{18}$ comporte 62 bits. Il faut faire attention aux erreurs d'arrondi !

On utilise maintenant la formule :

$$X(i+1) = a \times X(i) \bmod m = [(X(i) \times a_1 \bmod m) \times a_0] \bmod m$$

- recherche de l'inverse *b* et factorisation

On recherche l'inverse de a modulo m. C'est un nombre b tel que $a \times b = 1 \bmod m$. Ce qui permettra de parcourir l'ensemble des $<X_n>$ "à l'envers".

On utilise le petit théorème de Fermat :

$$a^{m-1} = 1 \bmod m, \qquad \text{et donc} \qquad b = a^{m-2} \bmod m.$$

la valeur de b obtenue est la seule valeur de l'inverse qui convienne. C'est à dire :

$$b = 721609879133 = 11 \times 13 \times 17 \times 17 \times 23 \times 759173 = 950521 \times 759173 = b_0 \times b_1$$

Notons que la factorisation de b est beaucoup plus favorable que celle de a, puisque $9.51\ 10^{17}$ comporte seulement 60 bits. On pourra utiliser le même algorithme de calcul dans les deux cas.

- écriture du programme

```
function random : double ;

begin
  asm
    fild     qword ptr[mm]        { ld m }
    fild     qword ptr[xx]        { X(i) }
    fimul    dword ptr[c1]        { X * a1 }
    fprem                         { (X * a1) mod m }
    fimul    dword ptr[c0]        { ((X * a1) mod m) * a0 }
    fprem                         { (((X * a1) mod m) * a0) mod m }
    fld      st                   { ld X }
    fistp    qword ptr[xx]        { sto X(i+1) }
    fmul     qword ptr[im]        { X / m }
    fst      qword ptr[bp - 08]   { retour résultat }
    finit
  end ;
end ;

function random_(cc : longint) : longint ;

begin
  asm
    fldcw    cwd                  { arrondi par troncature }
    fild     qword ptr[mm]        { ld m }
    fild     qword ptr[xx]        { X(i) }
    fimul    dword ptr[c1]        { X * a1 }
    fprem                         { (X * a1) mod m }
    fimul    dword ptr[c0]        { ((X * a1) mod m) * a0 }
    fprem                         { (((X * a1) mod m) * a0) mod m }
    fld      st                   { ld X }
    fistp    qword ptr[xx]        { sto X(i+1) }
    fmul     qword ptr[im]        { X / m }
    fimul    dword ptr[bp + 06]   { * n (cc) }
    fist     dword ptr[bp - 04]   { retour résultat }
    finit
  end ;
end ;
```

Remarques :

La valeur de l'inverse de *m* est mémorisée pour éviter la division de normalisation du résultat. La valeur de *X* entière est enregistrée, tandis que la valeur du résultat flottant est retournée par la fonction. **c1** et **c0** sont les copies de a_1 et a_0, ou b_1 et b_0, suivant les cas.

- Recherche d'une répartition uniforme de petites graines

Dans une première approche j'ai calculé **toutes** les puissances de *a* (en 17 jours), ce qui m'a permis de remplir le tableau suivant. Sont présentés dans ce tableau les exposants *p* et les valeurs X_p de a^p mod *m*, qui sont inférieures ou égales à 181.

Recherche des exposants qui produisent de petits valeurs (d:\tp\random\rempli.pas)

X_p	p	X_p	p	X_p	p	X_p	p
2	329683503893	3	637064608199	4	659367007786	5	932120680362
6	966748112092	7	670558002880	8	989050511679	9	274129216410
10	261804184267	11	141650211240	12	296431615997	13	216913063584
14	241506785	15	569185288573	16	318734015584	17	693340895560
18	603812720303	19	383693275699	20	591487688160	21	307622611091
22	471333715133	23	58099779320	24	626115119890	25	864241360736
26	546596567477	27	911193824609	28	329925010678	29	424545256310
30	898868792466	31	227585387701	32	648417519477	33	778714819439
34	23024399465	35	602678683254	36	933496224196	37	468401678624
38	713376779592	39	853977671783	40	921171192053	41	690872424613
42	637306114984	43	772940079894	44	801017219026	45	206249896784
46	387783283213	47	626181083908	48	955798623783	49	341116005772
50	193924864641	51	330405503771	52	876280071370	53	219343754009
54	240877328514	55	73770891614	56	659608514571	57	20757883910
58	754228760203	59	358786339599	60	228552296371	61	141439285096
62	557268891594	63	944687219290	64	978101023370	65	149033743958
66	108398323344	67	841302088733	68	352707903358	69	695164387519
70	932362187147	71	39790792781	72	263179728101	73	541844124006
74	798085182517	75	501305968947	76	43060283497	77	812208214120
78	183661175688	79	324237215774	80	250854695958	81	548258432820
82	20555928518	83	340278868061	84	966989618877	85	625461575934
86	102623583799	87	61609864521	88	130700722931	89	863907247918
90	535933400677	91	887471066464	92	717466787106	93	864649995900
94	955864587801	95	315813956073	96	285482127688	97	902459070119
98	670799509665	99	415779427650	100	523608368534	101	651247480439
102	660089007664	103	74072781694	104	205963575275	105	239743291465
106	549027257902	107	348032195601	108	570560832407	109	121297969747
110	403454395507	111	105466286835	112	989292018464	113	15463699117
114	350441387803	115	990220459682	116	83912264108	117	491042279994
118	688469843492	119	363898898452	120	558235800264	121	283300422480
122	471122788989	123	327937032824	124	886952395487	125	796362041110
126	274370723195	127	620520747490	128	307784527275	129	410004688105
130	478717247851	131	664949293799	132	438081827237	133	54251278591
134	170985592638	135	843314504984	136	682391407251	137	195026869325
138	24847891424	139	824838677233	140	262045691052	141	263245692119
142	369474296674	143	358563274824	144	592863231994	145	356665936684
146	871527627899	147	978180613971	148	127768686422	149	730242239133
150	830989472840	151	515081819013	152	372743787390	153	967470111970
154	141891718025	155	159706068075	156	513344679581	157	506427825916
158	653920719667	159	856408362208	160	580538199851	161	728657782200
162	877941936714	163	758255907216	164	350239432411	165	710835499813
166	669962371954	167	179794558877	168	296673122782	169	433826127168
170	955145079827	171	657822492109	172	432307087692	173	175119429695
174	391293368414	175	534799363628	176	460384226824	177	995850947798
178	193590751823	179	292674704201	180	865616904570	181	267701937061

Après réflexion, je n'ai calculé que les termes de la forme :

$$a^p \bmod m = g \qquad \text{avec } p = k\,(m \operatorname{div} 32) \pm e$$

et déterminé la valeur de l'écart *e* qui permettait d'obtenir une petite valeur de la graine.

Valeur des exposants et écarts pour 32 points de départ (intervalles 3.125e10) de la fonction RANDOM / nombres inférieurs à 2^16 (d:\tp\test\random\rech.pas)

k	Graine *g*	Écart *e*	Exposant *p*	Écart relatif
1	55338	3094097	31253094097	0.00010
2	20927	7292478	62507292478	0.00023
3	15461	4703549	93754703549	0.00015
4	52168	1431151	125001431151	0.00005
5	54531	2208489	156252208489	0.00007
6	11513	901736	187500901736	0.00003
7	53024	1917759	218751917759	0.00006
8	24290	-3160527	249996839473	-0.00010
9	23742	-5078686	281244921314	-0.00016
10	45189	11961276	312511961276	0.00038
11	65016	-7580902	343742419098	-0.00024
12	47601	6329140	375006329140	0.00020
13	17139	8141735	406258141735	0.00026
14	28255	-14573884	437485426116	-0.00047
15	23589	-4992796	468745007204	-0.00016
16	57450	127371	500000127371	0.00000
17	6556	9458171	531259458171	0.00030
18	48661	-4023895	562495976105	-0.00013
19	26454	2450952	593752450952	0.00008
20	46911	-25684495	624974315505	-0.00082
21	60670	2590170	656252590170	0.00008
22	60114	3773632	687503773632	0.00012
23	19056	4269209	718754269209	0.00014
24	35558	17632396	750017632396	0.00056
25	37554	3099744	781253099744	0.00010
26	21679	7185807	812507185807	0.00023
27	7956	183352	843750183352	0.00001
28	10277	20412496	875020412496	0.00065
29	12840	17995865	906267995865	0.00058
30	28368	11509208	937511509208	0.00037
31	52439	3714738	968753714738	0.00012
32	1	-12	999999999988	-0.00000

Valeur des exposants et écarts pour 32 points de départ (intervalles 3.125e10) de la fonction Random / produits de facteurs inférieurs à 184 (d:\tp\test\random\recherch.pas)

k	Graine g	Écart e	Exposant p	Écart relatif
1	23108	8731554	8731554	0.00028
2	22630	16304014	31233695986	0.00052
3	30843	28765125	62471234875	0.00092
4	4095	29036740	93720963260	0.00093
5	19276	43508668	125043508668	0.00139
6	54531	2208489	156252208489	0.00007
7	21750	155409537	187655409537	0.00497
8	57552	3195218	218746804782	0.00010
9	726	48534584	250048534584	0.00155
10	54510	44212428	281205787572	0.00141
11	13630	30524497	312530524497	0.00098
12	65016	7580901	343742419099	0.00024
13	47601	6329141	375006329141	0.00020
14	21321	51777424	406301777424	0.00166
15	36295	41136078	437458863922	0.00132
16	11475	26080930	468776080930	0.00083
17	8723	2559920	500002559920	0.00008
18	6556	9458171	531259458171	0.00030
19	31958	32707776	562467292224	0.00105
20	4784	3141512	593746858488	0.00010
21	30229	35034229	625035034229	0.00112
22	35371	149297809	656399297809	0.00478
23	34771	6562703	687493437297	0.00021
24	39458	66589299	718683410701	0.00213
25	10266	79708013	750079708013	0.00255
26	5542	30306681	781280306681	0.00097
27	21679	7185807	812507185807	0.00023
28	7956	183352	843750183352	0.00001
29	31122	22937522	874977062478	0.00073
30	12840	17995865	906267995865	0.00058
31	27145	32786612	937467213388	0.00105
32	24013	88229463	968838229463	0.00282
33	24047	41893072	999958106928	0.00134

Notons que la connaissance de a et b et des 32 points de départ permet de disposer de 64 séquences indépendantes (pour le jeu d'échec), de longueur de l'ordre de 3.125×10^{10}. C'est à dire de 64 séquences permettant de tester environ 1 milliard de trajets chacune (34.7 cases parcourues en moyenne).

VII. EXPLORATION DE L'ARBRE DES SOLUTIONS

Première famille de tests

Points de départ équirépartis. Ces statistiques portent sur 17.9×10^9 essais, 2983 parcours complets ayant été réalisés.

Nombre moyen de branches en fonction de la profondeur

Départ	5.25	4.98	5.06	4.86	4.83	4.66	4.68
4.52	4.52	4.36	4.36	4.20	4.20	4.04	4.04
3.88	3.88	3.72	3.72	3.56	3.56	3.40	3.40
3.24	3.24	3.08	3.08	2.92	2.92	2.76	2.76
2.59	2.59	2.43	2.43	2.26	2.26	2.10	2.10
1.93	1.93	1.77	1.77	1.60	1.60	1.44	1.43
1.27	1.27	1.10	1.10	0.94	0.94	0.78	0.77
0.62	0.61	0.45	0.45	0.30	0.29	0.14	0.13

Résultats associés par paires, on remarque que les cases noires et les cases blanches sont indépendantes. La longueur moyenne de parcours est de 34.6 cases.

Largeur moyenne de l'arbre en fonction de la profondeur

Départ	5.25e00	2.62e01	1.33e02	6.44e02	3.11e03	1.45e04	6.78e04
3.06e05	1.38e06	6.02e06	2.62e07	1.10e08	4.63e08	1.87e09	7.55e09
2.93e10	1.14e11	4.23e11	1.58e12	5.61e12	2.00e13	6.80e13	2.31e14
7.50e14	2.43e15	7.48e15	2.30e16	6.72e16	1.96e17	5.40e17	1.49e18
3.86e18	1.00e19	2.43e19	5.90e19	1.34e20	3.02e20	6.35e20	1.33e21
2.58e21	4.98e21	8.80e21	1.55e22	2.49e22	3.98e22	5.72e22	8.20e22
1.04e23	1.32e23	1.46e23	1.60e23	1.51e23	1.41e23	1.10e23	8.50e22
5.24e22	3.20e22	1.46e22	6.54e21	1.93e21	5.55e20	7.88e19	1.04e19

La largeur de l'arbre de recherche augmente jusqu'à la $52^{\text{ème}}$ case. Puis le nombre de branches disponibles devient inférieur à 1.

Inverse de la probabilité de réaliser le parcours jusqu'à la case *N*

1.000	1.000	1.000	1.000	1.001	1.001	1.003	1.004	**8**
1.007	1.010	1.014	1.018	1.024	1.029	1.037	1.045	**16**
1.055	1.065	1.079	1.092	1.110	1.128	1.151	1.174	**24**
1.203	1.233	1.272	1.311	1.362	1.413	1.480	1.550	**32**
1.640	1.734	1.858	1.990	2.164	2.352	2.606	2.885	**40**
3.270	3.702	4.314	5.023	6.057	7.299	9.188	11.56	**48**
15.35	20.39	28.99	41.22	63.99	99.37	173.1	302.0	**56**
615.5	1259.	3215.	8288.0	30120.	112500	793100	6017000	**64**

Deuxième famille de tests

Points de départ spécifique. Ces statistiques portent sur 11.6×10^9 essais, 1874 parcours complets ayant été réalisés.
Points de départ équirépartis. *Ces statistiques portent sur $17.9 \acute{} 10^9$ essais, 2983 parcours complets ayant été réalisés.*

Largeur de l'arbre $\times 10^{22}$

18.1 ± 0.8			946 ± 75 (*1027 ± 19*)
18.8 ± 1.7	17.8 ± 1.5		
17.6 ± 1.3	13.3 ± 1.2	9.9 ± 0.9	
16.9 ± 1.3	13.4 ± 1.1	10.3 ± 1.0	9.9 ± 0.7

Taille de l'arbre (Nombre de solutions $\times$ Nombre d'essais = Nombre de feuilles) $\times 10^{25}$

5.16 ± 0.01			367 ± 1 (*400 ± 1*)
6.75 ± 0.01	7.68 ± 0.01		
7.02 ± 0.01	5.76 ± 0.05	4.41 ± 0.03	
5.98 ± 0.02	5.80 ± 0.04	4.20 ± 0.02	3.55 ± 0.03

Difficulté : les deux résultats obtenus pour l'ensemble de l'échiquier sont incompatibles. (La somme des produits des moyennes n'est pas égale au produit des sommes des moyennes ???)

Nombre de parcours $\times 10^{18}$

28.5 ± 1.3			596 ± 44 (*665 ± 12*)
10.0 ± 0.9	9.2 ± 0.8		
11.2 ± 0.8	3.0 ± 0.3	4.9 ± 0.5	
13.2 ± 1.0	5.6 ± 0.5	6.1 ± 0.6	8.1 ± 0.6

Dans le bilan relatif aux 64 cases de l'échiquier, on compte chaque parcours dans les deux sens. Le nombre de parcours est de l'ordre de 64×10^{19}.

Troisième famille de tests

Ces statistiques portent sur 6.0×10^{10} essais, 10732 parcours complets ayant été réalisés.

Nombres de parcours réussis

2800			10732
1385	692		
848	483	814	
1447	505	478	1280

Nombres d'essais par réussite $\times 10^6$

2.00 ± 0.04			7.60 ± 0.29
5.51 ± 0.15	8.66 ± 0.33		
6.90 ± 0.24	14.79 ± 0.67	9.14 ± 0.32	
4.53 ± 0.12	9.69 ± 0.43	7.40 ± 0.34	4.21 ± 0.12

Nombre moyen de réussites par essais $1/[(6.00 \pm 0.06) \times 10^6]$

Points de départ et d'arrivée : résultats symétrisés (symétries du carré, symétrie des solutions)

32.3 ± 0.4 %			100.0 %
12.5 ± 0.2 %	6.5 ± 0.2 %		
9.0 ± 0.2 %	3.0 ± 0.1 %	3.9 ± 0.1 %	
12.8 ± 0.2 %	5.2 ± 0.2 %	5.4 ± 0.2 %	9.4 ± 0.2 %

A comparer au nombre de parcours obtenu précédemment :

28.6 ± 1.3 %			100.0 %
10.0 ± 0.9 %	9.2 ± 0.8 %		
11.2 ± 0.8 %	3.0 ± 0.3 %	4.9 ± 0.5 %	
13.2 ± 1.0 %	5.7 ± 0.5 %	6.1 ± 0.6 %	8.1 ± 0.6 %

Chemins fermés

Ces statistiques portent sur 6.0×10^{10} essais, 1091 parcours fermés ayant été réalisés.

Nombres de parcours fermés réussis

77			1091
102	75		
79	146	237	
87	87	90	111

Nombres d'essais par réussite $\times 10^{7}$

7.3 ± 0.8			6.1 ± 0.6
7.5 ± 0.7	8.0 ± 0.9		
7.4 ± 0.8	4.9 ± 0.4	3.1 ± 0.2	
7.5 ± 0.8	5.6 ± 0.6	3.9 ± 0.4	4.9 ± 0.5

Le nombre de parcours fermés est indépendant de la case de départ, puisque tous les parcours fermés passent sur toutes les cases. On obtient le plus de succès avec la case 3×3.

Pourcentage des parcours réussis qui sont fermés

2.8 ± 0.4 %			10.2 ± 0.4 %
7.4 ± 0.9 %	10.8 ± 1.7 %		
9.3 ± 1.4 %	30.2 ± 3.9 %	29.1 ± 2.9 %	
6.0 ± 0.8 %	17.2 ± 2.6 %	18.8 ± 2.8 %	8.7 ± 1.1 %

Nombre de solutions $\times 10^{17}$

7.9 ± 0.4			9.5 ± 0.7 (*10.6 ± 0.7*)
7.3 ± 0.7	10.0 ± 1.0		
10.4 ± 0.9	9.2 ± 1.2	14.2 ± 1.7	
7.9 ± 0.7	9.7 ± 1.0	11.5 ± 1.4	7.1 ± 0.6

Le nombre de parcours fermés est indépendant de la case de départ, puisque tous les parcours fermés passent sur toutes les cases. Le nombre de parcours fermés distincts est de l'ordre de 10^{18}.

VIII. PROBLÈMES ANNEXES

I) Élimination des branches mortes de l'arbre

Comment éliminer les branches mortes de l'arbre de recherche complet ? En réalisant un test de "connexité" de l'ensemble des cases qui n'ont pas été atteintes (test identique à l'algorithme de reconstruction du pentomino proposé par Rouben Ter Minassian).

La meilleure efficacité a été obtenue en contrôlant la connexité au niveau des noeuds de profondeur 41 et 45. Ceci ne permet pas toutefois de trouver des solutions plus sûrement, lorsque l'on n'en trouve pas.

II) Entre deux cases

- plus court chemin

- nombre de solutions minimales

III) Variantes du problème

- échiquiers de différentes tailles

- échiquiers de différentes formes (rectangle, ruban de möbius, tore, cube ...)

IX. BIBLIOGRAPHIE

Pour la Science / Septembre 1987 / Jeux Mathématiques / Ian Stewart

Error Control Coding - Fundamentals and Applications / Shu Lin / Daniel J. Costello, Jr / Prentice Hall

The Art of Computer Programming / Volume 2, Third Edition - Random Numbers / Donald E. Knuth / Addison Wesley

Maple Tech / Volume1, Number 1, Spring 1994 / Random Number Generation and Testing / Zaven A. Karian and Rohit Goyal / Birkhäuser