LYCEE NEVERS, MONTPELLIER

TPE 2017-2018

FABRIQUER DU HASARD

1° S3

LISA BAGET
ARTHUR JACQUIN
LUJZA LEHOCKA
LOUNA MIGUEL

ENCADRANTS

MADAME MANTE
MADAME PALOC
MADAME SOMBARD

THEMES

MATHEMATIQUES, BIOLOGIE

« DIEU NE JOUE PAS AUX DES ».

ALBERT EINSTEIN, 1927

IMAGE DE COUVERTURE : STATUE DE FORTUNA, DEESSE DU HASARD. MYTH & LEGENDS COLLECTION, 2018.

TABLE DES MATIERES

INTRODUCTION

Le hasard est présent au quotidien dans notre vie. La sécurité des comptes bancaires, nos téléphones portables dépendent de codes qui utilisent une grande quantité de nombres tirés au hasard. Si ce hasard est de mauvaise qualité, les codes seront également mauvais et ceci facilitera le piratage. Nous ne nous sommes pas intéressés à ces codes mais plutôt à ce hasard qui est aujourd'hui omniprésent.

Pourtant, au XIXe siècle, le hasard avait disparu des sciences (Marchal, 2004). Le mathématicien Pierre-Simon de Laplace écrivait en 1814 :

« Nous devons envisager l'état de l'Univers comme l'effet de son état antérieur et la cause de ce qui va suivre. Une intelligence qui pour un instant donné connaîtrait toutes les forces dont la nature est animée et la situation respective des êtres qui la composent, […] embrasserait dans la même formule le mouvement des plus grands corps de l'Univers et ceux du plus léger atome : rien ne serait incertain pour elle, l'avenir comme le passé serait présent à ses yeux. »

Ce déterminisme laplacien enlève toute place au hasard. Par exemple le tirage d'une pièce à pile ou face n'est pas déterminé par le hasard mais par des lois physiques. Ce n'est qu'au début du XXème siècle que revient vraiment le hasard en sciences. Henri Poincaré écrit en 1908:

« Une cause très petite, qui nous échappe, détermine un effet considérable que nous ne pouvons pas ne pas voir, et alors nous disons que cet effet est dû au hasard...Mais, lors même que les lois naturelles n'auraient plus de secret pour nous, nous ne pourrons connaître la situation initiale qu'approximativement. Si cela nous permet de prévoir la situation ultérieure avec la même approximation, c'est tout ce qu'il nous faut, nous dirons que le phénomène a été prévu, qu'il est régi par des lois ; mais il n'en est pas toujours ainsi, il peut arriver que de petites différences dans les conditions initiales en engendrent de très grandes dans les phénomènes finaux... »

Si on veut prévoir le résultat d'un tirage à pile ou face, il faut des mesures infiniment précises, ce qui est impossible. Une minuscule erreur de mesure peut changer le résultat prévu. Même si le tirage est déterministe, son issue est imprévisible. En physique quantique la grande majorité des chercheurs s'accordent à dire que le hasard intervient au niveau de l'atome. Einstein n'y croyait pas et affirmait en 1927 « *Dieu ne joue pas aux dés* ». Pourtant, le mouvement des atomes serait bien soumis à des probabilités et le fait que le mouvement de la pièce soit déterministe est lié à la loi des grands nombres.

Si le hasard n'existe qu'au niveau de l'atome mais que le reste est entièrement déterminé par la loi des grands nombres (même si les imprécisions ou erreurs sur les mesures des paramètres font que le résultat peut être imprévisible), comment font les scientifiques pour *fabriquer du hasard* ? Comment peuvent-ils affirmer qu'il est de bonne qualité (ou de suffisamment bonne qualité pour sécuriser les codes bancaires) ? Est-il même possible de fabriquer du hasard ou de le mesurer ?

Dans la première partie, nous découvrirons les travaux de Jean-Jacques Kupiec, qui a découvert de nouvelles sources de hasard en biologie moléculaire. Il défend aujourd'hui la théorie du Darwinisme cellulaire.

Dans une seconde partie, nous verrons comment les mathématiciens mesurent la qualité du hasard, et comment les informaticiens en fabriquent (ou font semblant d'en fabriquer)

Enfin, dans la troisième partie, nous réaliserons nos propres expériences pour savoir si l'être humain est capable de fabriquer ou bien reconnaitre du hasard.

1 LA PRESENCE DU HASARD EN GENETIQUE

Les scientifiques considèrent depuis longtemps que le fonctionnement des êtres vivants est entièrement déterminé : nous avons déjà parlé du déterminisme laplacien, et, au XVIIème siècle, Descartes utilise la métaphore de la machine pour expliquer le fonctionnement des êtres vivants. Encore en 1944, le physicien Erwin Schrödinger défend le déterminisme en biologie moléculaire, et dénie toute place au hasard dans le fonctionnement d'une cellule. Il considère que, même si chaque particule d'un système est soumise au hasard, la loi des grands nombres fait que le fonctionnement d'une cellule est entièrement déterministe.

Cette idée est attaquée par Jean-Jacques Kupiec, biologiste et épistémologue au Centre Cavaillès de l'Ecole Normale Supérieure de Paris, qui élabore une nouvelle théorie : le darwinisme cellulaire (Kupiec, 2009).

1.1 INTRODUCTION AU DARWINISME CELLULAIRE

Le Darwinisme cellulaire provient de la théorie de Darwin qui affirme que l'homme descend du singe dans son ouvrage *L'origine des espèces* publié en 1859. Darwin affirme qu'au cours du temps, les êtres vivants ont subi des mutations (sans en connaitre les mécanismes exacts), c'est à dire une modification de l'ADN. De plus il développe sa théorie en expliquant qu'il existe un mécanisme qui permet d'éliminer des organismes (êtres vivants) dits « moins adaptés à leur environnement » : c'est la sélection naturelle.

Dans la théorie du Darwinisme cellulaire les gènes sont soumis à des mécanismes probabilistes qui permettent aux cellules de se différencier de manière aléatoire. Le développement embryonnaire ne serait pas déterminé par un programme génétique mais par la sélection naturelle de notre organisme.

Nous nous appuierons sur les travaux et expériences de Jean-Jacques Kupiec. Depuis les années 1980 il défend la théorie du darwinisme cellulaire. Grâce à ces travaux, nous essaierons de comprendre où, aujourd'hui, les chercheurs voient du hasard dans le vivant.

Figure 1 Molécule d'ADN

Nous étudierons donc l'expression génétique (de l'information héréditaire du gène à la fabrication de molécules) à travers différents exemples.

1.2 LA DIFFERENCIATION CELLULAIRE

Selon la théorie dominante les cellules sont génétiquement programmées et leur spécialité serait donc prédéfinie. Jean-Jacques Kupiec considère au contraire que la cellule embryonnaire se développe de manière aléatoire pour définir sa spécialité (Kupiec, 2013). Ce processus se nomme la différenciation cellulaire.

Au début, les cellules possèdent toutes le même génome (ensemble des gènes), elles n'en exprimeraient qu'une partie et vont donc acquérir une identité par les gènes qu'elles expriment (soit en neurone, en cellules musculaires, etc…). Ce processus sélectif des gènes est le produit de plusieurs niveaux de régulation de l'expression génétique. L'un des processus majeurs de régulation est assuré par les protéines régulatrices qui se fixent directement sur l'ADN au niveau des gènes.

1.2.1 Création d'une protéine

Nous avons appris cette année que les protéines ouvrent la double hélice de l'ADN pour permettre la transcription. C'est donc à partir de cette séquence que la transcription est initiée. Après l'ouverture de la double hélice, la protéine qui s'appelle l'ARN polymérase associe les nucléotides libres (adenine, uracile, cytosine et guanine) aux nucléotides dans le noyau (adenine, thymine, cytosine et guanine) qui forment l'ADN, par complémentarité avec le brin transcrit de l'ADN. Puis l'ARN sort du noyau et se dirige vers le cytoplasme où la transcription est suivie par la traduction. La traduction correspond à une autre phase qui associe au brin de l'ARN messager un ARN transfert qui est constitué de trois nucléotides qui codent les différents acides aminés.

1.2.2 Protéine responsable de la différenciation cellulaire

Selon la théorie dominante, les protéines régulatrices se déplacent dans l'ordre d'un gène à l'autre, empêchant tout hasard. Nous proposerons une explication simplifiée des schémas ci-dessus.

Imaginons trois gènes positionnés à proximité de l'un de l'autre mais à des distances non équivalentes. Le gène 1 (G1) entraîne la différenciation des cellules embryonnaire en globule rouge, le gène 2 (G2) entraîne la différenciation des cellules embryonnaires en neurones et enfin, le gène 3 (G3) entraîne la différenciation des cellules embryonnaires en cellules musculaires.

La protéine régulatrice se trouve au niveau de G1 et interagit une première fois avant de se détacher. Elle se déplace ensuite au hasard le long de l'ADN avant de se fixer à nouveau de manière aléatoire.

Cependant, il y a plus de probabilité que cette protéine régulatrice vienne se refixer sur G1 qu'avec les autres gènes. Si la protéine régulatrice se trouve au niveau de G2, elle a plus de chance de se déplacer sur G3 qui est plus proche que G1, là encore on retrouve un hasard mais tout de même lié à la proximité des gènes environnant la protéine.

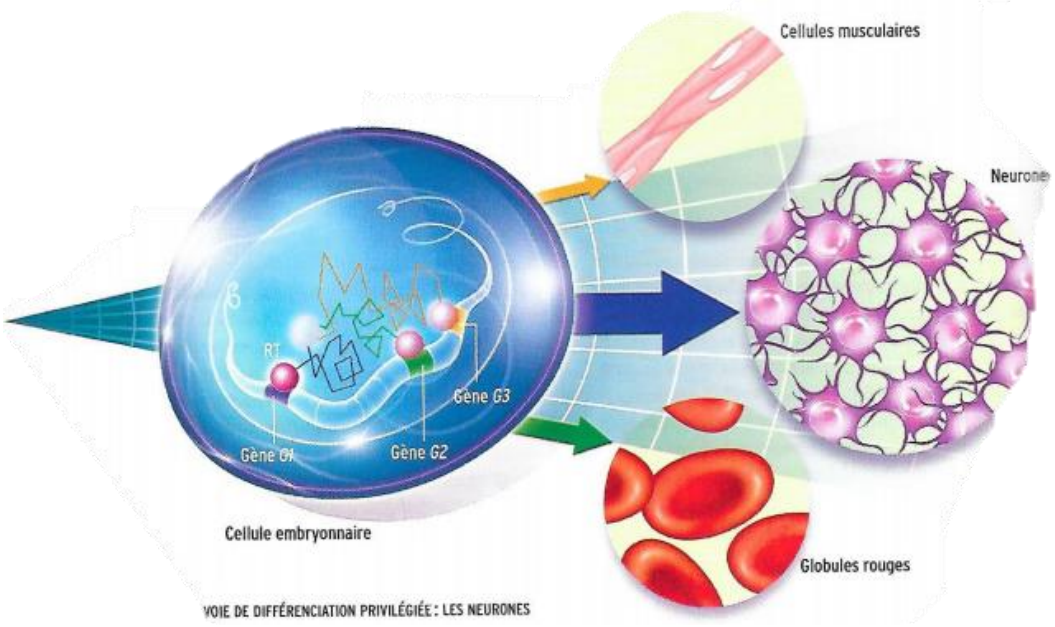


Figure 2 Voix de différenciation privilégiée: les neurones

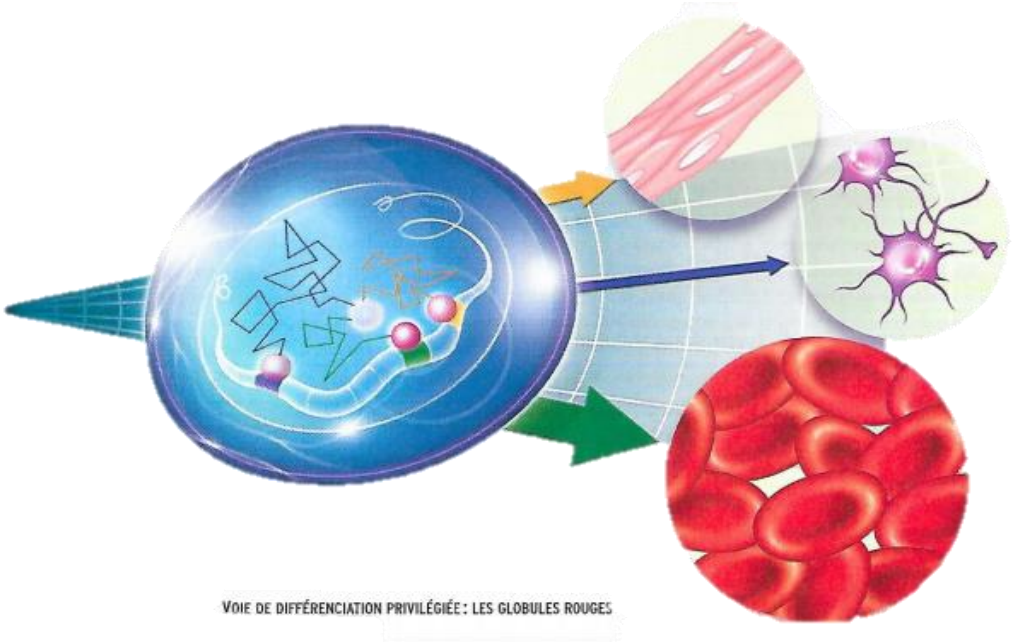


Figure 3 Voie de différenciation privilégiée: les globules rouges

D'après Kupiec la protéine se déplace de manière aléatoire autour de la molécule d'ADN et va se fixer sur un gène au hasard. Cette action déterminera la spécialité de cette cellule. Bien que dans l'organisme les chercheurs aient trouvé plusieurs cas ou le hasard n'a pas sa place, ici Jean-Jacques Kupiec trouve que la différenciation cellulaire fait intervenir le hasard (Chevassus-au-Louis, 2017). Il se peut cependant que des explications encore méconnues des chercheurs existent.

1.3 L'ASSEMBLAGE DES MOLECULES

Pour autre exemple nous pouvons nous appuyer sur le fonctionnement moléculaire.

1.3.1 Structure d'une molécule

Selon la théorie dominante, les molécules comme les protéines ont une structure tridimensionnelle, ce qui leur permet de s'emboîter selon un système clef qui exclut le hasard (enzyme substrat). Il existe donc un nombre limité d'interactions possibles entre ces molécules.

1.3.2 Divers assemblages possibles

Dans la démonstration de la théorie de Jean-Jacques Kupiec nous avons observé que les protéines peuvent interagir avec un grand nombre molécules différentes, dépendant de leur rencontre aléatoire que nous avons expliqué par les deux figures précédentes. Les molécules peuvent adopter une conformation précise en fonction de la molécule « partenaire » avec laquelle elles interagissent. Cependant le choix du « partenaire » est aléatoire et dépend donc des rencontres entre ces molécules. Beaucoup de protéines contiennent des régions désordonnées incapables de s'organiser en une structure tridimensionnelle fixe. Ces régions se stabilisent grâce à l'interaction avec une autre molécule. Cette plasticité confère à ces protéines de nombreux partenaires potentiels et donc plusieurs fonctions. La cartographie des interactions protéiques dans les organismes entiers montre qu'au moins 10% des protéines peuvent interagir avec plus de 100 partenaires. Le hasard est appliqué dans le choix de ces partenaires, ce hasard au niveau moléculaire existe, mais le nombre des molécules impliquées est tellement grand que la variabilité globale du système devient négligeable. De plus, les événements aléatoires au niveau moléculaire débouchent sur des comportements reproductibles au

niveau de l'organisme. Même si le hasard au niveau des molécules n'est pas très important, cette théorie probabiliste nous permet à expliquer plusieurs phénomènes. Par exemple, la théorie déterministe n'est pas capable d'expliquer que 95% de l'ADN ne code aucun gène, ou que des phases de mortalité massive participent au développement, car ce n'est pas logique que l'évolution conserve de l'ADN qui n'est pas indispensable ainsi que des cellules vouées à mourir. Au contraire, si ces processus sont aléatoires, c'est normal qu'il y ait des échecs.

La théorie de Kupiec nous présente la possibilité d'un hasard au sein de la protéine régulatrice. Cette théorie a été approuvée par de nombreux scientifiques croyant en l'aléatoire dans le domaine biologique. La théorie pose toujours des doutes mais Jean-Jacques Kupiec étudie le darwinisme cellulaire et défend la variabilité dans la différenciation cellulaire. Deux cellules d'une même lignée ne sont jamais strictement identiques. Comme il l'écrit :

« *On a toujours considéré que cette variabilité était du bruit que l'on délaissait. Dorénavant, elle doit être étudiée pour elle-même, en tant que paramètre fondamental du vivant.* »

2 FABRIQUER DU HASARD EN INFORMATIQUE

Un ordinateur ne fait rien au hasard. Il est composé de circuits, et les valeurs de sortie sont entièrement déterminées par les valeurs d'entrée. Pourtant, on a l'impression qu'il donne parfois des résultats aléatoires, par exemple quand nous utilisons la fonction *aleat* sur notre calculatrice, ou quand, dans un jeu vidéo, la même action ne produit pas toujours le même résultat. L'ordinateur fait donc semblant de fabriquer du hasard. Pour ceci, les mathématiciens ont dû d'abord définir ce qui est du « bon hasard » (2.1), puis les informaticiens ont écrit des algorithmes pour le produire (2.2). Enfin, nous verrons que les ordinateurs utilisent aussi aujourd'hui la nature pour produire du hasard (2.3).

Figure 4 Dilbert, par Scott Adams

2.1 EVALUER LA QUALITE DU HASARD

Nous avons vu en 3° que *« une expérience aléatoire est une expérience dont on ne peut pas prédire le résultat ».* En probabilité, on suppose qu'il existe un hasard parfait (« une pièce non truquée », « un dé parfaitement équilibré »), pourtant nous avons vu que l'existence de ce hasard était débattu. Nous nous posons une autre question: en observant les résultats d'une expérience, nous nous demandons si ces résultats peuvent provenir du hasard.

Nous considérons dans la suite du TPE que le résultat d'une expérience est une suite de '0' et de '1'. Par exemple '0111' veut dire qu'on a successivement tiré pile, puis 3 fois face. Nous appelons cette séquence une *chaîne*, puisque c'est pour un ordinateur une chaîne de caractères binaires. Examinons maintenant la chaîne '00000000000000000000000000000000'. Il ne semble à personne que c'est le résultat d'une expérience aléatoire. Pourtant, sa probabilité d'apparaître est la même que pour toutes les autres chaînes de même longueur. Comment alors définir si une chaîne est aléatoire ?

2.1.1 Définitions du hasard

Les mathématiciens du XXe siècle ont cherché à définir ce qu'est une chaîne aléatoire (Wikipedia, 2009). Mais, dans les années 1930, le mathématicien français Emile Borel a affirmé, par un paradoxe, que le hasard est indéfinissable. Il commence par affirmer qu'une chaîne aléatoire est une chaîne sans caractéristique particulière. Mais le fait de ne pas avoir de caractéristique est déjà une caractéristique.

En 1966, le logicien suédois Per Martin-Löf définit une chaîne aléatoire comme une chaîne n'ayant aucune caractéristique particulière *« effectivement testable »*, c'est-à-dire vérifiable par un algorithme. Il limite les caractéristiques et évite ainsi le paradoxe de Borel.

En 1971, le mathématicien allemand Claus-Peter Schnorr définit une chaîne aléatoire comme une chaîne contre laquelle aucune martingale n'est possible. Il voit ceci comme un jeu : si, grâce à qu'il a déjà observé, il peut prédire (en utilisant un algorithme) le chiffre suivant avec plus d'une chance sur deux, alors il peut gagner de l'argent en pariant sur le résultat, et la chaîne n'est donc pas au hasard.

En 1974, l'américain Gregory Chaitin et le russe Leonid Levin utilisent les travaux du russe Andreï Kolmogorov pour proposer une autre définition. Une chaîne aléatoire est une chaîne *incompressible*, c'est-à-dire qu'il n'existe pas d'algorithme plus petit que la chaîne qui permet de générer cette chaîne. Comme les trois définitions sont équivalentes, nous nous concentrons sur celle-ci.

2.1.2 Complexité de Kolmogorov

La définition de Levin et Chaitlin repose sur la complexité de Kolmogorov (Wikipedia, 2005), qui mesure la quantité de hasard dans une chaîne. Prenons par exemple la chaîne de taille 200.000

'0101010101010101010101010101010101010101010101010101010101010101 … 0101010101010101'

Un algorithme permettant de générer cette chaîne est simplement celui qui dit de l'afficher. La taille de cet algorithme (nombre de caractères pour l'écrire) est 200000+8. Mais cette chaîne peut également être générée par l'algorithme « *Répéter 100000 fois '01'* », dont la taille (le nombre de caractères dont on a besoin pour l'écrire) est 25. Comme la taille de ce programme est plus petite que la taille de la chaîne, la chaîne n'est pas aléatoire.

> **Définition :** *La complexité de Kolmogorov d'une chaîne est la taille du plus petit programme qui génère cette chaîne.*

La complexité de Kolmogorov mesure la quantité de hasard d'une chaîne. Quand la complexité d'une chaîne est égale à sa taille, ça veut dire que la chaîne est aléatoire. Nous voulions utiliser cette complexité pour notre TPE : il nous suffisait de trouver un programme calculant cette complexité pour mesurer le hasard. Malheureusement, à cause d'un théorème, ce programme ne peut pas exister.

> **Théorème :** *La complexité de Kolmogorov n'est pas calculable.*

Il n'est donc pas possible d'écrire un algorithme qui calcule cette complexité pour n'importe quelle chaîne. C'est pour ça que nous n'avons pas pu trouver un tel programme sur internet.

2.1.3 De multiples critères

S'il est impossible de mesurer la quantité de hasard d'une chaîne, les chercheurs ont créé des tests afin de déterminer si une chaîne n'est pas aléatoire. Lorsqu'une chaîne réussit tous ces tests, ça ne veut pas dire qu'elle est aléatoire, mais qu'aucun test n'a encore été inventé pour la faire échouer. Le NIST américain (National Institute of Standards and Technology) utilise 15 tests différents pour garantir la qualité d'un générateur aléatoire (Barker, et al., 2016). Ici, nous expliquerons deux de ces tests :

Frequency (Monobits) Test Ce test est en fait un test de de moyenne. Considérons la chaîne

'00000000000000000000000000000000100000000000000000000000000000000000000000000000'

La moyenne des valeurs prises par cette chaîne de taille 80 est (0x79 + 1x1)/80 = 1/80 = 0,0125. Or, si cette chaîne avait été obtenue de façon aléatoire, l'espérance serait de 0,5. Plus la chaîne est grande, plus la probabilité d'obtenir un tel écart entre la moyenne et l'espérance est faible, et donc plus la probabilité que cette chaîne est aléatoire est faible.

Test For Frequency Within A Block Ce deuxième test est aussi appelé test de *k*-uniformité. Expliquons d'abord ce qu'est un test de 2-uniformité. On considère la chaîne :

'00000000000000000000000000000000000000001111111111111111111111111111111111111111'

constituée de 40 '0' suivis de 40 '1'. La moyenne est cette fois-ci de 0,5 et donc cette chaîne va réussir le test de la moyenne (frequency monobits test). C'est normal, car la moyenne s'intéresse au nombre d'apparitions de '0' et de '1', mais pas à leur position. On regarde maintenant tous les blocs de taille 2, c'est-à-dire les 79 paires de termes. On trouve 39 fois '00', une fois '01', 39 fois '11', et jamais '10'. Or si cette chaîne était aléatoire, ces 4 paires possibles seraient équiprobables. Comme cette chaîne est assez grande, et que l'écart entre le nombre d'apparition des blocs et ce qui est prévu par les probabilités est grand, cette chaîne a très peu de chances d'être aléatoire.

Cependant, comme la 2-uniformité ne regarde que des blocs de taille 2, on peut quand même tromper ce test. Par exemple, on considère la chaîne

‘00110011001100110011001100110011001100110011001100110011001100110011001100110011’

On trouve 20 fois ‘00’, 20 fois ‘11’, 20 fois ‘01’ et 19 fois ‘10’. Cette chaîne réussit donc le test de 2-uniformité, mais ce n’est bien évidemment pas une chaîne aléatoire, puisque elle est obtenue en répétant 20 fois ‘0011’. On ne sera pas trompé en prenant des blocs de taille plus grande : dans un test de *k*-uniformité, on compte les blocs différents de taille *k*. Plus *k* est grand, plus on a de chances de repérer qu’une chaîne n’est pas aléatoire.

2.2 PRODUIRE DU HASARD PAR UN ALGORITHME

Nous allons maintenant tenter de comprendre comment fonctionne un générateur aléatoire.

2.2.1 Principes de base

Un générateur aléatoire est en fait une suite définie par récurrence (Wikipedia, 2005). On se donne un entier u_0 appelé la *graine*, et la suite est définie par $u_{n+1} = f(u_n)$ (chaque terme est défini en fonction du terme précédent), où f est une fonction de $\mathbb{N}^+$ dans $\mathbb{N}^+$.

Par exemple, prenons $u_0 = 12$ et $f(n) = 2n$.

$u_1 = f(u_0) = f(12) = 24$; $u_2 = f(u_1) = f(24) = 48$; $u_3 = f(u_2) = f(48) = 96$;…

A chaque fois qu’un programme utilise une fonction qui a besoin d’un nombre aléatoire, le générateur aléatoire produit le terme suivant de la suite, et utilise ce terme pour calculer ce dont a besoin le programme.

Le problème de la période. Supposons que les premiers termes de notre générateur aléatoire soient : $u_0 = 5$, $u_1 = 7$, $u_2 = 12$, $u_3 = 15$, $u_4 = 4$, $u_5 = 12$. On remarque que $u_2 = u_5 = 12$. Donc $u_6 = f(u_5) = f(u_2) = u_3 = 15$. De même $u_7 = f(u_6) = f(u_3) = u_4 = 4$ et $u_8 = f(u_7) = f(u_4) = u_2 = 12$. Les nombres générés seront donc 5, 7, 12, 15, 4, 12, 15, 4, 12, 15, 4, 12, 15, 4, 12, …

Nous avons écrit en vert les termes jusqu’à ce qu’il y ait une répétition. Jusqu’ici, tout se passe bien… Mais le premier terme rouge est un nombre que nous avions déjà trouvé. A partir de là, on ne peut que répéter la séquence de termes entre le premier 12 (vert) et le second (rouge). La suite n’est pas aléatoire puisqu’on peut la compresser par *Ecrire ‘5, 7, 12, 15, 4’, puis répéter ’12, 15, 4’*. Or ceci arrivera forcément : puisque dans un ordinateur il y a une valeur maximale pour les entiers, il y aura forcément une répétition à un moment donné. On appelle *période* la taille de la séquence (en vert) avant qu’il y ait une répétition. Ici la période est de 5. Une bonne caractéristique pour un générateur aléatoire sera d’avoir une période la plus grande possible (pour maintenir l’illusion du hasard le plus longtemps possible). Bien entendu, la période dépend de la graine et de la fonction f utilisée.

2.2.2 La méthode de Von Neumann

Von Neumann est un mathématicien allemand. Il met au point les premiers ordinateurs et participe au programme nucléaire. Il a aussi inventé, en 1946, le premier générateur aléatoire reposant sur un algorithme. C’est la méthode des carrés médians (Wikipedia, 2004).

Le générateur de Von Neumann génère des nombres de taille N, c’est-à-dire qu’on peut écrire avec N chiffres entre 0 et 9. Si N = 6, la valeur maximale générée est 999999. Maintenant, supposons que le générateur de Von Neumann ait généré le nombre u_k. Nous allons expliquer ce que fait la fonction f qui génère u_{k+1}.

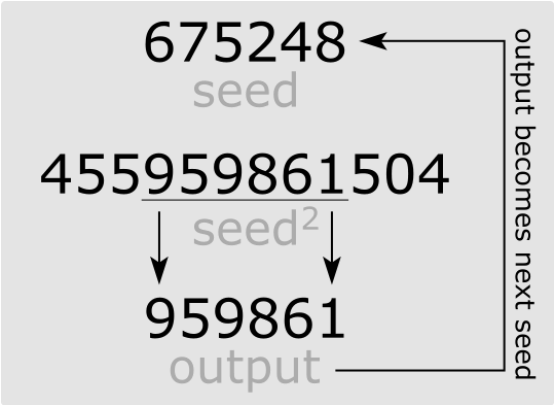


Figure 5 Etape de l'algorithme de Von Neumann

Dans cet exemple, on a u_k = 675248. On met ce nombre au carré et on obtient 455959861204 qui est un nombre trop grand, puisqu'il a 12 chiffres. On sélectionne alors les 6 chiffres du milieu pour obtenir u_{k+1} = 959861. On calculera u_{k+2} à partir de u_{k+1} de la même manière.

Le problème de cette méthode est que la période est trop courte (surtout si on choisit mal la graine). Cet algorithme a été amélioré par de nombreuses méthodes toujours en application de nos jours.

2.2.3 Les méthodes modernes

En 1948, Derrick Lehmer introduit une nouvelle famille de générateurs aléatoires, les générateurs congruentiels linéaires ou GCL (Wikipedia, 2005). Ces générateurs sont encore utilisés aujourd'hui dans de nombreux logiciels, même si ils peuvent être mauvais quand on choisit mal les paramètres (Gillespie, 2016). Ils sont peu à peu remplacés par de meilleurs générateurs comme le Mersenne Twister qui a de meilleurs scores aux tests du NIST.

Par exemple, le tableur EXCEL utilise un GCL (Pollock, 2014), notre calculatrice TI-82 mélange les résultats de deux GCLs (Inconnu, 2015), et le langage de programmation PHP propose un GCL et, depuis la version 4 (PHP, 2005), également un Mersenne Twister qui est recommandé si on souhaite du bon hasard.

2.3 FABRIQUER DU HASARD SANS ALGORITHME

Une chaîne générée par un algorithme n'est évidemment pas aléatoire : sa complexité de Kolmogorov est inférieure à la taille de l'algorithme. Si on juge que c'est « du bon hasard », c'est qu'on n'a pas encore inventé le test permettant de prouver que ce n'est pas du hasard. C'est une course sans fin entre de meilleurs générateurs aléatoires et de meilleurs tests. Et si la solution n'était pas algorithmique ?

2.3.1 Chercher le hasard dans la nature

Imaginons qu'un humain jette des pièces parfaitement équilibrées à pile ou face, de manière à ce que le résultat soit imprévisible. S'il notait tous ses résultats sous forme d'une chaîne de '0' et de '1', obtiendrait-il une chaîne parfaitement aléatoire ? C'est ce que suppose la théorie des probabilités. Mais le problème est qu'une telle méthode ne nous permet pas de fabriquer beaucoup de hasard rapidement. Pour produire une chaine de taille 1 million, il faudrait au moins 1 million de secondes, c'est-à-dire plus de 11 jours. Or les applications informatiques (sécurité bancaire, GPS, …) ont aujourd'hui besoin de milliards et de milliards de bits aléatoires chaque jour.

2.3.2 Fabriquer beaucoup de hasard

Même si l'idée d'utiliser la nature pour générer du hasard semble bonne, il faut trouver d'autres méthodes pour fabriquer beaucoup de hasard. Parmi toutes les idées proposées, on peut par exemple

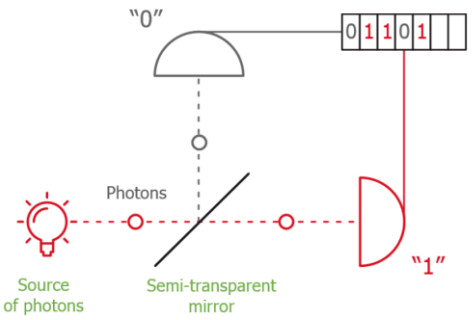


Figure 6 Chaîne générée par physique quantique

citer la mesure des variations du bruit atmosphérique (utilisée par le générateur du site https://www.random.org/integers/).

Une idée récente est d'utiliser les propriétés aléatoires de la physique quantique. Dans le schéma ci-contre, des photons sont envoyés un par un (très rapidement) sur un miroir semi-transparent incliné à 45 degrés. La physique quantique nous dit (mais ne nous demandez pas pourquoi) que chaque photon a une chance sur deux de continuer sa trajectoire, et une chance sur deux d'être dévié de 90 degrés par rapport à sa trajectoire initiale. Dans tous les cas, un capteur repère l'arrivée du photon, note 0 si il a été dévié et 1 sinon. Une telle expérience nous donne 1 bit aléatoire par photon envoyé.

2.3.3 Un générateur quantique dans l'ordinateur

Figure 7 Carte Quantis QRNG PCIe 16Mb

Aujourd'hui, la société IDQ, par exemple, propose des cartes que l'on peut brancher sur la carte mère de n'importe quel ordinateur, qui contiennent un micro-appareil qui réalise l'expérience que nous avons décrite. Dans leur brochure (SA, 2017) ils affirment que cette carte, qui coûte quand même 3000€, peut produire une chaîne parfaitement aléatoire de taille 16.000.000 en une seconde. Les chaînes produites réussissent tous les tests du NIST et peuvent donc être considérées, en l'état actuel des connaissances, comme parfaitement aléatoires.

Si on n'a pas une telle carte, un site comme http://www.randomnumbers.info/ permet de récupérer gratuitement des chaînes aléatoires qu'elle produit. Ce sont ces chaînes aléatoires que nous allons utiliser lors des expériences de notre dernière partie.

3 HASARD ET CERVEAU HUMAIN

L'être humain a tendance à attribuer au hasard les évènements. «J'ai eu 15 amendes pour excès de vitesse cette année, quelle malchance», dira un automobiliste imprudent. D'un autre côté, l'être humain a aussi tendance à trouver des explications à ce qui est purement du hasard : « à chaque fois que j'oublie mon parapluie, il pleut ». Nous utilisons des expériences menées par des informaticiens pour savoir si l'être humain est capable de reconnaitre si une expérience est au hasard ou non (3.1), puis des expériences relatées dans deux articles de J.-P. Delahaye pour savoir si il peut lui-même produire du hasard de bonne qualité (3.2). Enfin, nous nous inspirons de tous ces travaux pour mener nos propres expériences (3.3).

3.1 RECONNAISSANCE DES REGULARITES

Nous avons vu au chapitre précédent qu'il est difficile d'écrire un programme reconnaissant si une chaîne n'est pas aléatoire. Nous allons voir que le cerveau humain est parfois utilisé comme un test très efficace pour reconnaitre les chaînes non aléatoires.

3.1.1 Représenter graphiquement une chaîne aléatoire

Afin d'expliquer les expériences qui ont été réalisées par des chercheurs, nous avons conçu un petit exemple qui montre comment on peut représenter, par une image, une chaîne. Considérons la chaîne de taille 121 suivante :

'10011111001001001001000111010111010100100101100011100011111101111110001110001
10100100101011101011100010010010010011111001'

Il est difficile, en la regardant, de savoir si cette chaîne est aléatoire ou pas. Mais on peut mettre ces chiffres dans un tableau de taille 11x11. Chaque case est coloriée en noir si le chiffre est 1, et est coloriée en blanc si le chiffre est 0.

1	0	0	1	1	1	1	1	0	0	1
0	0	1	0	0	1	0	0	1	0	0
0	1	1	1	0	1	0	1	1	1	0
1	0	1	0	0	1	0	0	1	0	1
1	0	0	0	1	1	1	0	0	0	1
1	1	1	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	0	0	0	1
1	0	1	0	0	1	0	0	1	0	1
0	1	1	1	0	1	0	1	1	1	0
0	0	1	0	0	1	0	0	1	0	0
1	0	0	1	1	1	1	1	0	0	1

Tableau 1 Chaîne organisée dans un tableau

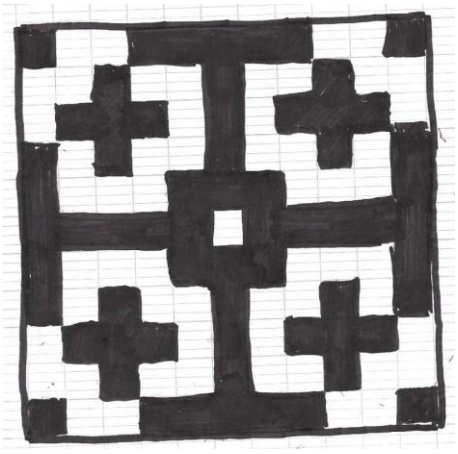


Figure 8 Représentation graphique de la chaîne

On remarque que le dessin obtenu, une croix de Jérusalem, n'est vraiment pas aléatoire. Le cerveau traite l'image immédiatement et reconnait des régularités. Pourtant, nous avons conçu cet exemple (en rajoutant du noir dans les coins, en créant un trou au milieu, …) pour tromper les tests statistiques. En effet, il y a dans cette chaîne 64 apparitions de '1' et 57 apparitions de '0', ce qui est dans la marge d'erreur acceptée par le *Frequency (Monobits) Test*. De plus, il y 27 apparitions de '00', 31 de '01', 30 de '10' et 32 de '11', ce qui est aussi dans la marge d'erreur acceptée par le *Test For Frequency Within A Block* (en se limitant à des blocs de taille 2). On voit grâce à notre petit exemple que le cerveau humain peut percevoir des régularités qui ne sont pas repérées par les tests (les plus simples) du NIST.

3.1.2 Utilisation de la vision pour évaluer des générateurs aléatoires

Une méthode similaire est utilisée pour évaluer les chaînes générées par différents générateurs aléatoires (Inconnu, 2015). Dans cet article de blog, l'auteur considère des nombres entre 0 et N^2 produits par différents générateurs aléatoires. A chacun de ces nombres correspond un point placé dans une image de taille NxN. Chaque fois qu'un nombre apparait, on fonce le point qui lui correspond. Il génère ainsi des images qui illustrent le tirage de très nombreux nombres aléatoires.

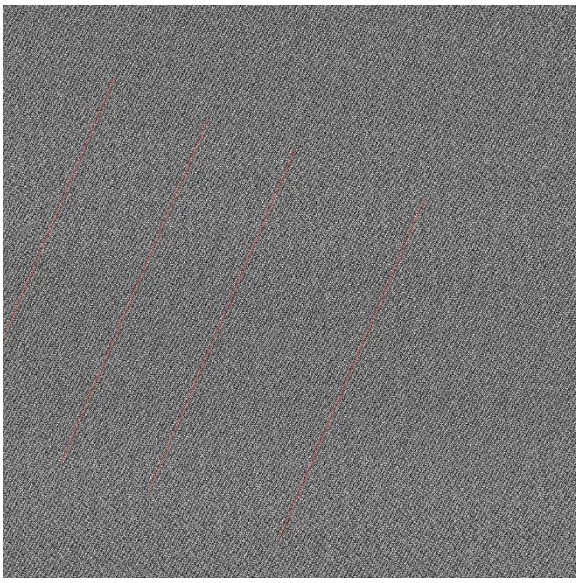


Figure 9 Représentation de nombres générés par Ocaml

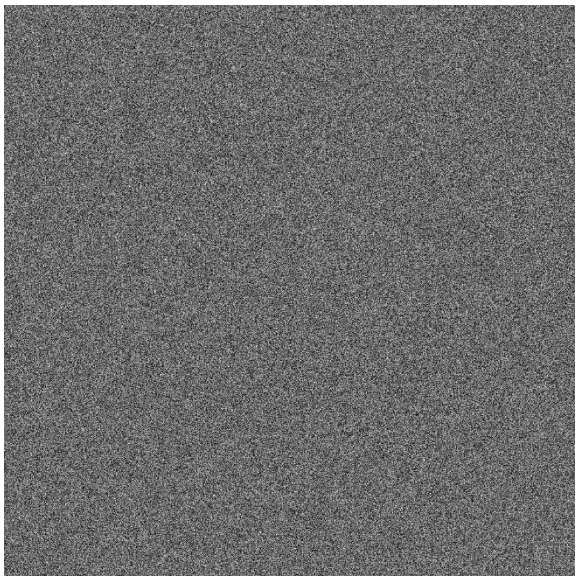


Figure 10 Représentation de nombres générés par Fortuna

L'image de gauche utilise cette technique pour montrer que le générateur aléatoire utilisé dans le langage OCaml (un GCL) n'est pas de très bonne qualité. En effet, nous voyons des bandes diagonales plus foncées qui se répètent régulièrement (l'auteur les a soulignées en rouge, et on les voit encore mieux sur les images du site web). Dans l'image de droite, il utilise des nombres générés par le programme Fortuna (d'après la déesse grecque du hasard). Nous ne percevons aucune régularité.

Le cerveau humain peut ainsi repérer grâce à la vision des régularités qui seraient très difficiles à repérer par des tests statistiques. C'est ce qui permet aux experts de concevoir de nouveaux tests qui détecteront ces régularités perçues intuitivement par notre cerveau.

3.1.3 Autres pistes

Nous venons de voir que, en choisissant une bonne méthode de représentation graphique pour des données aléatoires, la vision et le cerveau humain permettaient de percevoir intuitivement des régularités qui indiquent clairement un mauvais hasard. D'autres sens nous permettraient-ils d'arriver au même résultat ? Dans la page https://www.random.org/audio-noise/, une séquence aléatoire est utilisée pour produire une représentation sonore de ce hasard. Il n'est malheureusement pas possible d'utiliser ce programme avec un mauvais hasard pour vérifier si nous pouvons entendre les régularités.

3.2 LE CERVEAU PRODUIT-IL DU BON HASARD ?

Pour répondre à cette question, nous avons lu deux articles de la revue *Pour la Science* écrits par Jean-Paul Delahaye, professeur d'informatique à l'université de Lille.

3.2.1 Production de chaînes aléatoires

Dans (Delahaye, 2004), une expérience consiste à demander à des sujets de générer une chaîne de '0' et de '1' comme si c'était au hasard. D'après ces expériences, le résultat montre à peu près le même nombre de '0' et de '1'. Pourtant, cette expérience montre que les sujets produisent beaucoup trop

d'alternances, c'est-à-dire de '1' suivis de '0' ou de '0' suivis de '1'. Ceci veut exactement dire que le test de 2-uniformité donnera trop de '01' et de '10'. Dans cette expérience, il y a 60% d'alternances et 40% de répétitions, alors que la proportion attendue est de 50-50. Or il y a très peu de chances qu'une chaîne aléatoire vérifie un tel écart (0,28% avec une chaîne de taille 200).

Cette expérience montre que le cerveau humain produit du mauvais hasard. Une autre expérience consiste à demander à un sujet A si le hasard produit par B est de bonne qualité. Le résultat est que la « reconnaissance du hasard » est également mauvaise, ce qui contredit apparemment ce que nous avons dit précédemment, mais cette expérience n'a pas utilisé d'image pour aider la reconnaissance.

3.2.2 Le hasard géométrique

L'article (Delahaye, et al., 2006) décrit des expériences menées par le chercheur canadien John Christie. Dans une première expérience, on demande à 600 personnes de placer chacun 3 points au hasard dans un carré, puis on superpose dans une même image les 1800 points obtenus.

Le résultat, représenté ci-contre, montre une grande symétrie. Les sujets ont privilégié le centre du carré et certains axes (diagonales, axe vertical principal). Tout se passe comme si, lorsqu'on place au hasard un point dans une figure géométrique, on privilégie les zones qui correspondent à des propriétés mathématiques bien connues.

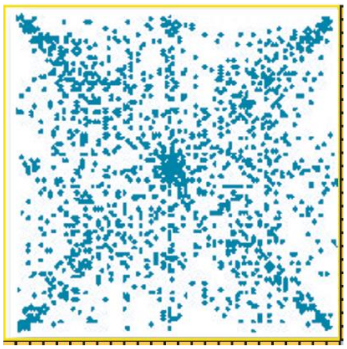


Figure 11 1800 points au hasard

3.2.3 Les biais cognitifs

Toujours dans (Delahaye, 2004), des expériences ont montré que la production de hasard par le cerveau humain était faussée par des biais cognitifs. Par exemple, lorsqu'on demande au hasard à un sujet un chiffre entre 0 et 9 (chacun a une probabilité de 10%), le chiffre 7 apparaît dans 30% des cas. Il semble être jugé plus aléatoire que les autres, car on lui associe moins de propriétés mathématiques évidentes. Ceci est paradoxal car pour Christie (3.2.2), ils favorisent les zones qui ont le plus de propriétés.

Un autre biais cognitif est que lorsqu'on donne le choix entre deux alternatives, le *biais de positivité* pousse les sujets à choisir en majorité celle qui semble positive : oui ou non donne oui à 62%, aimer ou détester donne aimer à 83%, indemne ou blessé donne indemne à 68%. Même lorsque les alternatives sont neutres, un autre biais intervient, le *biais de préséance* : les sujets choisissent majoritairement l'alternative qui a été présentée en premier. Lorsque l'on demande « pile ou face », « pile » sera choisi plus souvent car proposé en premier. De façon surprenante, ce biais de préséance semble plus marqué chez les hommes que chez les femmes.

3.3 NOS EXPERIENCES

Nous avons réalisé plusieurs expériences pour savoir si l'être humain peut produire et reconnaitre du hasard « de bonne qualité ». Pour la production de hasard, nous avons fait remplir des fiches à plus de 150 personnes et obtenu ainsi presque 10000 bits de données. Nous avons vite compris qu'il n'était pas possible d'effectuer des tests statistiques sérieux sans un traitement informatique (surtout que nous voulions aussi comparer ces résultats avec de vrais générateurs aléatoires). Comme nous n'avons aucune connaissance en programmation, nous avons cherché de l'aide. Nous remercions donc deux étudiants, Hugo Barral (images en PHP/SVG, formulaire HTML) et Hugo Piquard (statistiques en PHP). Nous leur faisions parvenir nos algorithmes, ils nous renvoyaient des programmes que nous testions sur de petites données, et recommencions souvent pour des corrections. Enfin, nous avons pu entrer les données de nos tests et générer les statistiques que nous présentons ici.

3.3.1 Expérience A : production de hasard

Notre première expérience est similaire à celle relatée par Delahaye (Delahaye, 2004), mais nous avions besoin de nos propres données comme élément de comparaison avec notre deuxième expérience. Nous avons demandé à plusieurs personnes (camarades de classe, famille, inconnus à la terrasse d'un Mac Donald) de remplir la fiche ci-dessous afin de créer une chaîne de taille 64. Nous avons reçu 92 réponses.

TPE – La fabrique du hasard – Baget, Jacquin, Lehocká, Miguel, 1ere S3– Lycée Nevers 2017-2018

Vous devez remplir chaque case du tableau ci-dessous avec un chiffre qui sera soit 0, soit 1 (un chiffre par case). Attention, chaque case devra être remplie « au hasard », comme si vous aviez tiré à pile ou face. Vous ne devez utiliser aucune source extérieure de hasard (ne tirez pas à pile ou face, ne générez pas de nombre avec un appareil électronique, …).

A

Frequency (monobit) test : Parmi les 92x64 = 5888 chiffres que nous avons générés, il y a 49,2% de '0' et 50,8% de '1'. Ce test est réussi, comme dans l'expérience de (Delahaye, 2004), et nous ne voyons pas non plus le biais de préséance en faveur des '0'. Pourtant, nous avions bien fait attention, dans nos consignes, de toujours citer le '0' en premier. Même quand on ne regarde que le premier chiffre répondu, 51% des résultats commencent par '0', ce qui n'est pas très significatif. L'article de Delahaye suggère que ce biais de préséance est peut-être lié au sexe. Malheureusement, nous n'avons gardé cette information que pour 23 des personnes qui ont passé le test (le prénom avait été inscrit sur la feuille). Sur les 14 hommes, 10 ont commencé par '0' (71,4%) et sur 9 femmes, 5 ont commencé par '0' (55,5%). Il semble donc que le biais de préséance est plus marqué chez les hommes que chez les femmes, mais nous n'avons pas réalisé assez d'expériences pour en être certain.

De plus, le nombre total de '0' et de '1', très proche de 50%, ne nous dit pas comment les personnes ont répondu individuellement. Il pourrait y avoir la moitié qui ont répondu beaucoup de '0', et la moitié qui ont répondu beaucoup de '1'. Nous avons donc construit deux séries statistiques dont les valeurs sont le nombre de '0' dans une chaîne, et avons calculé les fréquences (le pourcentage de personnes qui ont répondu cette valeur). La première série (en rouge dans l'histogramme ci-dessous) correspond aux résultats de notre expérience A. La seconde série (en bleu dans l'histogramme) correspond à 20000 créations de chaînes (également de taille 64) par le très bon générateur Mersenne Twister. Cette seconde série nous permet de comparer nos résultats avec ce qui est normalement attendu.

Que pouvons-nous dire de ces deux distributions? Les barres bleues, obtenues avec Mersenne Twister, indiquent une parfaite courbe en cloche centrée sur 32 (la moitié de 64). Cette courbe ressemble à celle qu'on devrait obtenir si on calculait les probabilités $P(X = x_i)$.

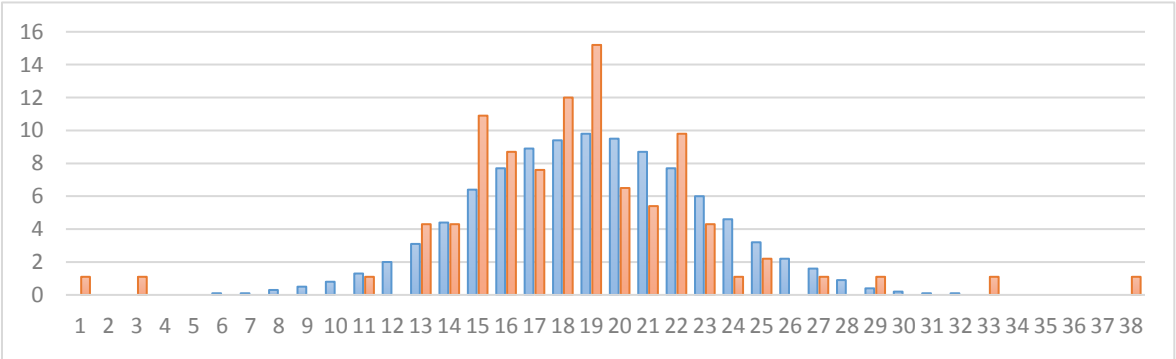


Figure 12 Distribution des '0': comparaison entre l'expérience A et Mersenne Twister

Les barres rouges, obtenues grâce à notre expérience, donnent un résultat assez différent. Tout d'abord, il y a 4 personnes chez qui le nombre de '0' est très anormal. Il se peut que ces personnes estiment très mal ce que doit être une chaîne aléatoire, mais il peut y avoir d'autres explications. Par exemple, une personne nous a dit « à la fin je n'ai mis que des 0 parce que j'étais pressé ».

Si on ignore ces résultats anormaux, on observe 3 pics : le premier est centré en 32, il y a autant de '0' que de '1'. Trop de personnes ont donné ce résultat « parfait » par rapport à ce qui est attendu dans la courbe bleue. Ceci se comprend car plusieurs personnes nous ont dit avoir essayé « de mettre autant de '0' que de '1' ». Les deux autres pics (à 28 et à 35 '0') montrent un déséquilibre: trop souvent, les personnes favorisent les '0' ou les '1'. Alors que le résultat moyen sur toutes les expériences était satisfaisant, nous identifions un groupe trop important qui favorise les '0' ou les '1'. Si nous avions gardé les informations sur les sexes, il aurait été intéressant de voir le lien avec le biais de préséance.

Test For Frequency Within A Block Nous avons calculé le nombre de chaînes de taille 2 différentes sur l'ensemble des réponses. Nous obtenons 22,9% de '00', 26,3% de '01', 26,3% de '10' et 24,5% de '11'. Ce résultat n'est pas très éloigné du résultat attendu (25% chacune), et si nous additionnons les '01' et '10' (c'est-à-dire si nous comptons les alternances), nous obtenons 52,6%. Nous observons un léger biais en faveur des alternances, même si il est moins important que celui rapporté par Delahaye (60%).

Comment comprendre cette différence avec les résultats de Delahaye ? Cela pourrait être dû à un échantillon trop petit, mais nous avons voulu regarder la distribution. Nous avons réalisé la même courbe que précédemment, mais en étudiant le nombre d'alternances (c'est-à-dire de '01' et de 10') au lieu du nombre de '0'. Dans l'histogramme, on voit apparaître une zone (entre 36 et 45) qui montre le biais d'alternance pour beaucoup de sujets. Mais la zone entre 16 et 21 montre des sujets qui ont créé un nombre anormalement bas d'alternances, ce sont eux qui font baisser la moyenne. En regardant les données, on voit que ce sont ceux qui ont créé de beaucoup trop longues successions de '0' ou de '1'. Sans ces résultats anormaux, notre moyenne serait plus proche de celle de Delahaye.

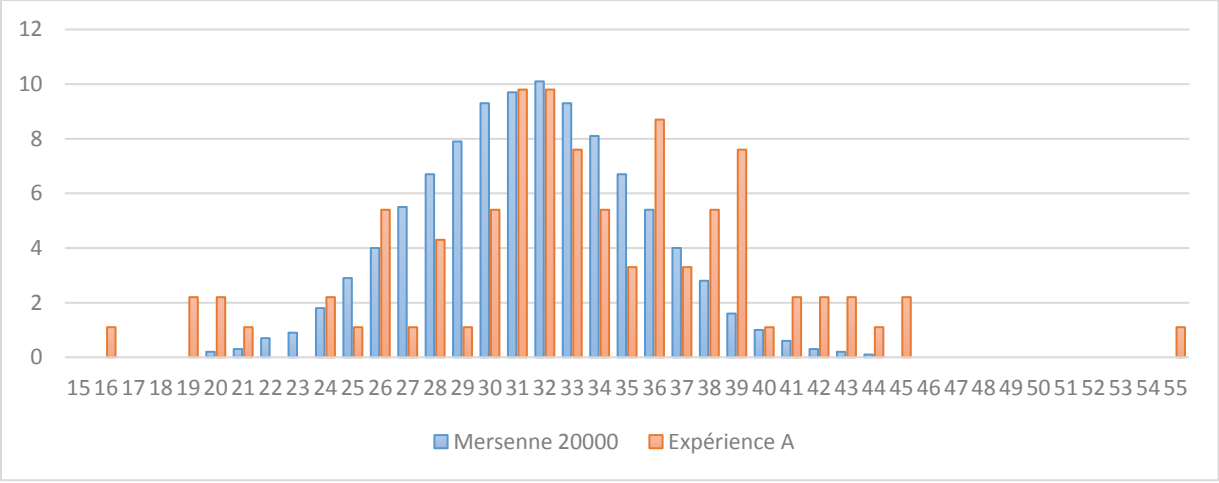


Figure 13 Distribution des alternances: comparaison entre l'expérience A et Mersenne Twister

3.3.2 Expérience B : production de hasard en utilisant un tableau

Comme il nous a semblé que l'être humain reconnait mieux le hasard graphiquement (quand les chaînes sont organisées dans un tableau), nous nous sommes demandé si on pouvait produire du hasard de meilleure qualité en l'organisant directement dans un tableau. Nous avons donc demandé à plusieurs personnes de remplir la fiche ci-dessous. Les sujets devaient, comme dans l'expérience A, remplir 64 bits, mais cette fois-ci ils étaient organisés dans un tableau 8x8. Nous avons eu 64 réponses.

TPE – La fabrique du hasard – Baget, Jacquin, Lehocká, Miguel, 1ere S3– Lycée Nevers 2017-2018

Vous devez remplir chaque case du tableau ci-dessous avec un chiffre qui sera soit 0, soit 1 (un chiffre par case). Attention, chaque case devra être remplie « au hasard », comme si vous aviez tiré à pile ou face. Vous ne devez utiliser aucune source extérieure de hasard (ne tirez pas à pile ou face, ne générez pas de nombre avec un appareil électronique, …).

B

Frequency (monobit) test : Dans cette expérience, il y a 48,8% de '0', ce qui est similaire au résultat de l'expérience A (49,2%). Ici aussi, nous étudions la distribution de ces '0'. Il y a encore un trop grand nombre de personnes qui ont trouvé exactement la moyenne, mais à part un sujet qui n'a trouvé que dix '0', on n'observe pas de résultats anormaux ni les deux pics qui indiquaient un déséquilibre. Le tableau semble avoir aidé les sujets de l'expérience à trop équilibrer le nombre de '0' et de '1'.

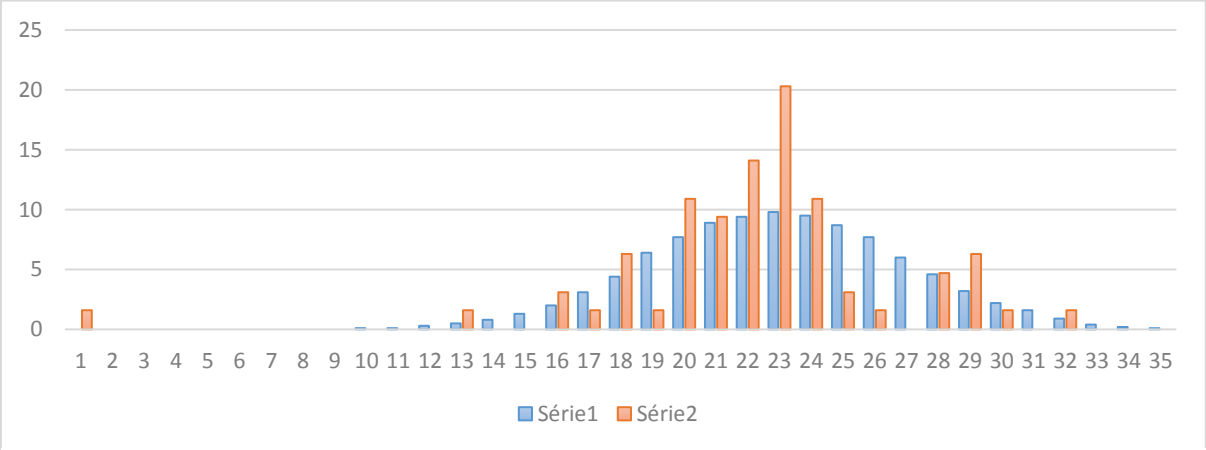


Figure 14 Distribution des '0': comparaison entre l'expérience B et Mersenne Twister

Test For Frequency Within A Block : Dans cette expérience, il y a 55,6% d'alternances, c'est-à-dire plus que ce que nous avions trouvé dans l'expérience A (52,6%), mais toujours moins que ce qui est dit par

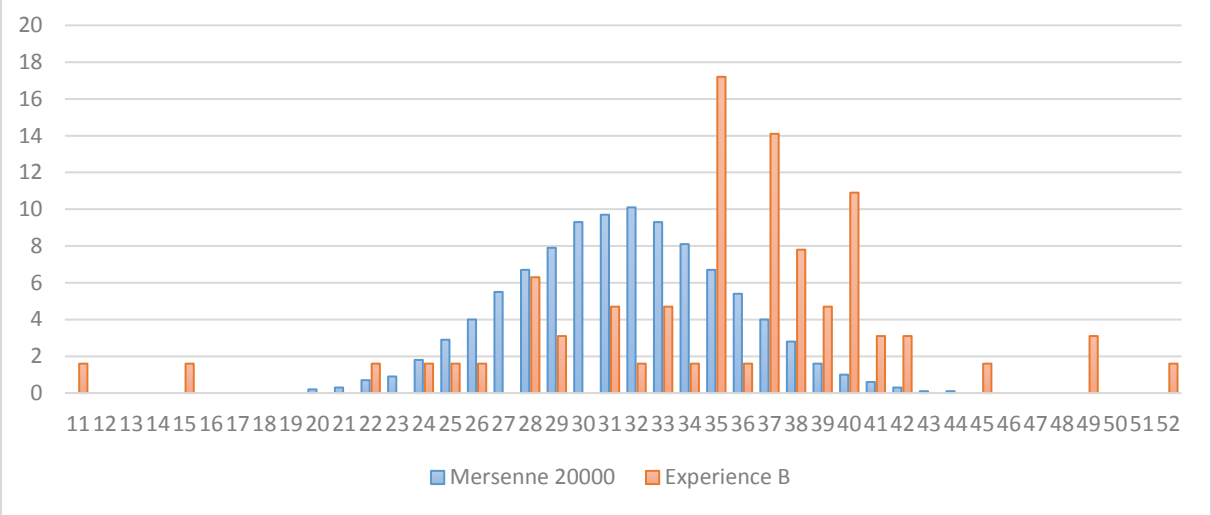


Figure 15 Distribution des alternances: comparaison entre l'expérience B et Mersenne Twister

Delahaye (60%). Contrairement à notre intuition, le biais d'alternance est encore plus marqué et l'organisation dans un tableau n'a pas aidé les sujets dans leur production de hasard. On voit dans l'histogramme que la zone qui indique le biais d'alternance (35-42) est un peu moins étalée que dans l'expérience A (36-45), mais beaucoup plus élevée (maximum à 17,2%, mais 8,7% dans l'expérience A). Nous expliquons ce renforcement du biais de la façon suivante : au lieu de remplir 2 grandes lignes de taille 32 dans l'expérience A, les sujets ont rempli 8 petites lignes de taille 8. Ceci a diminué le nombre et la taille de successions de bits identiques (par exemple, il est plus choquant d'avoir 4 '0' successifs dans une ligne de taille 8 que dans une ligne de taille 32), et donc mécaniquement augmenté le nombre d'alternances, ce qui a renforcé le biais.

Lignes et colonnes : Nous avons-nous même introduit un biais dans notre analyse des résultats. En effet, quand nous avons entré les données dans un fichier, nous avons lu les tableaux 8x8 ligne par ligne pour créer les chaînes que nous avons analysées. Pourtant, lorsque les personnes ont rempli ces tableaux, elles voyaient également les colonnes. Ceci peut faire une différence.

0	1	0	1
1	1	0	1
0	1	1	1
0	0	1	1

Tableau

0	1	0	1	1	1	0	1	0	1	1	1	0	0	1	1

Lecture du tableau ligne par ligne

0	1	0	0	1	1	1	0	0	0	1	1	1	1	1	1

Lecture du tableau colonne par colonne

Tableau 2 Différence entre la lecture d'un tableau ligne par ligne et sa lecture colonne par colonne

En lisant le tableau ligne par ligne, nous trouvons 9 alternances sur 15 (60%), mais en le lisant colonne par colonne, nous n'en trouvons plus que 5 (33%). Il faudrait maintenant calculer la distribution des alternances de notre expérience B en lisant chaque résultat en colonne, et comparer ceci avec la même lecture de l'expérience A pour savoir si l'organisation en tableau a amélioré ou empiré la qualité de hasard. Cette idée tardive n'a pas pu être présentée, les deux programmeurs préparant leurs examens.

3.3.3 Expérience C : reconnaissance du hasard

Nous avons conçu une expérience permettant de tester si un être humain est capable de reconnaitre du hasard. Cette expérience a pris la forme d'un sondage sur le site Survio.com, et notre questionnaire est visible à l'adresse https://www.survio.com/survey/d/L4P8F7M8E9W9J9O9M. Nous en avons fait la publicité sur Facebook, et nous avons reçu 92 réponses en 3 jours (900 personnes ont visité la page). La plupart des sondés ont pris entre 1 et 2 minutes pour répondre. Nous avons présenté plusieurs chaînes de '0' et de '1', soit sous forme **binaire** (les 324 premiers caractères de la chaîne elle-même), soit sous forme **graphique** (un dessin de la chaîne complète de taille 4096, comme expliqué en 3.1.1). A chaque fois, le sondé devait donner une note entre 1 (pas du hasard) et 5 (hasard parfait).

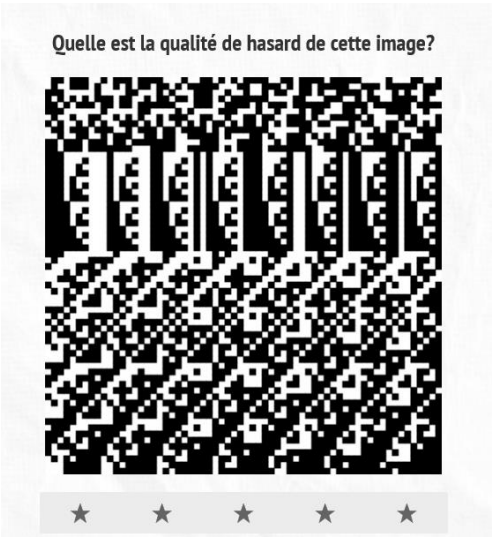


Figure 16 Notre sondage sur survio.com: évaluation du hasard sous forme binaire (ci-dessus) et sous forme graphique (à droite)

Type de hasard	Nom	Explication	Binaire	Graphique
Pas de hasard	D1	Répétitions de la chaîne '001'	1	x
	D2	Identique à D1, 8 '1' rajoutés	6	x
	D3	Enumération en binaire	16	13
Hasard humain	A1	64 premières réponses de l'expérience A	x	14
	A2	64 dernières réponses de l'expérience A	5	11
	B	Réponses de l'expérience B	7	15
Hasard informatique	G1	Générateur congruentiel linéaire RANDU	4	10
	G2	Calculatrice TI 82	3	9
	G3	Mersenne Twister	x	12
Hasard quantique	Q	Carte Quantis QRNG (voir 2.3.3)	2	8

Tableau 3 Elaboration des questions du sondage sur survio.com

Nous avons élaboré 10 chaînes de différentes façons. Trois ne sont pas au hasard : on doit le voir immédiatement sur D1, presque immédiatement sur D2, et c'est difficile à voir sur D3, qui est obtenue en collant la suite des entiers écrits en binaire (0', '1', '10', '11', '100', '101', '110', '111', '1000', …). Nous avons lu que cette chaîne D3 trompe tous les tests de k-uniformité, si elle est assez longue. Trois sont obtenues avec les résultats de nos expériences précédentes. Trois sont obtenues par des générateurs aléatoires : le très mauvais RANDU, notre calculatrice TI 82 (en générant des entiers entre 0 et 65535, on obtient 16 bits au hasard, il a fallu recommencer 4x64 fois), et le très bon Mersenne Twister. Enfin, nous avons récupéré sur le site http://www.randomnumbers.info/ 4096 bits générés par une carte quantique. Le tableau 3 résume ceci. La chaîne G2 sous forme binaire est notée G2b, elle est présentée à la question 3 du sondage. Sous forme graphique elle est notée G2g, et elle est présentée à la question 9.

Le site Survio.com permet de retirer des statistiques les sondés qui n'ont pas l'air sérieux. Nous en avons retiré un, le premier qui a répondu, et qui avait mis 5 étoiles pour toutes les questions. Les résultats que nous présentons maintenant ne concernent que les 91 autres sondés. Nous examinons dans un histogramme les notes moyennes (calculées par survio.com) pour les différentes chaînes.

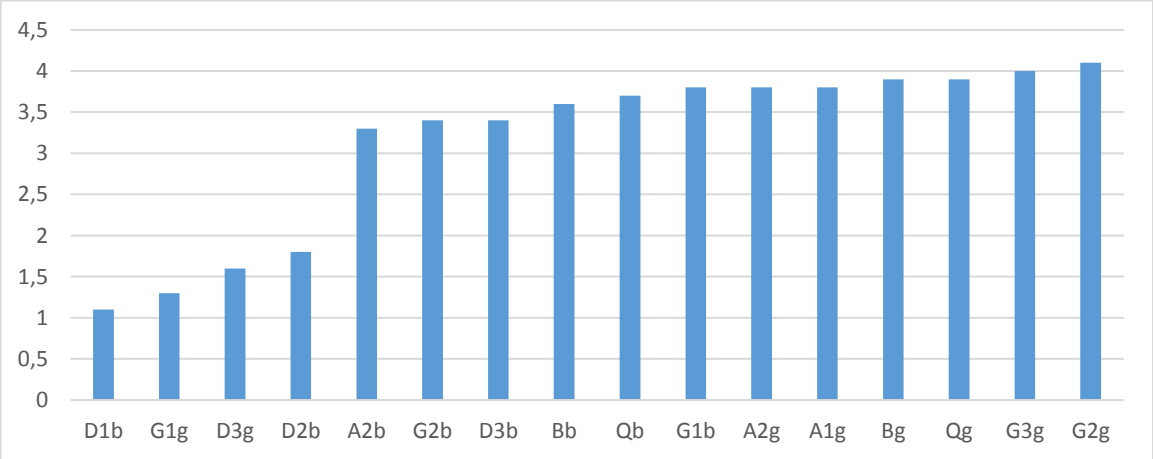


Figure 17 Notes moyennes obtenues par les différentes expériences

Nous observons dans cet histogramme deux groupes nettement séparés. Ceux qui ont une note inférieure à 1,8 et ceux qui ont une note supérieure à 3,3. Des chaînes ont été jugées bonnes, d'autres mauvaises, mais il n'y a rien entre les deux. Sans surprise, le mauvais groupe contient les chaînes binaires D1b et D2b, ainsi que les graphiques G1g et D3g. On remarque aussi que dans le bon groupe, les notes les plus basses sont toujours données aux chaînes binaires, et les plus hautes aux représentations graphiques.

Comme on le voit dans la figure suivante, l'ordre sur les représentations graphiques n'est pas surprenant, G1g et D3g sont clairement de moins bonne qualité que les autres. Nous voyons aussi que les chaînes générées par des personnes sont un peu moins bien notées que celles obtenues par générateurs, même si personne n'a pu nous l'expliquer. Seule surprise, la bonne note de Bg malgré la ligne noire horizontale suspecte au milieu (une trop longue séquence de '1').

Figure 18 Classement des chaînes présentées sous forme graphique

L'ordre sur les représentations binaires est plus surprenant. La chaîne G1b est la mieux notée des représentations binaires, alors qu'elle correspond à un générateur (RANDU) extrêmement mauvais, ce qui est prouvé par la très mauvaise note (justifiée) donnée à sa version graphique G1g. Nous allons donc comparer les notes attribuées aux mêmes chaînes sous leur forme graphique et binaire. On voit bien dans l'histogramme ci-dessous que les chaînes dont la forme graphique a été jugée mauvaise ont reçu de bonnes notes pour leur forme binaire. En fait, pour les moyennes des chaînes binaires, les sondés on soit vu que ce n'est pas du hasard (les cas simples D1 et D2), soit semblent n'avoir rien vu et donné des notes correctes (entre 3,3 et 3,8), mais pas aussi bonnes que pour les graphiques où ils ont « vu » que ça ressemble à du hasard (entre 3,8 et 4,1).

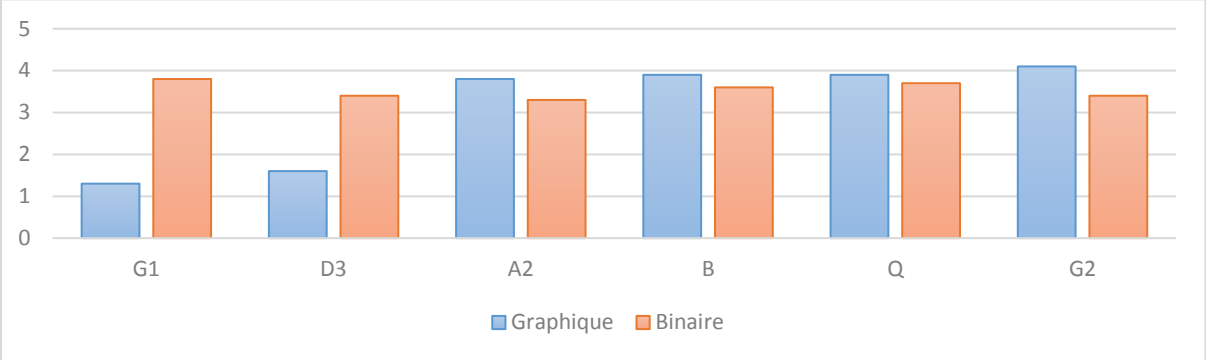


Figure 19 Comparaison des moyennes obtenues sous forme graphique et binaire

En conclusion, nous pouvons dire que :

1. A part dans les cas les plus simples (D1 et D2), les sondés sont incapables de reconnaitre du hasard quand il est présenté sous forme binaire. Ceci rejoint les conclusions de Delahaye.
2. La forme graphique aide fortement les sondés à reconnaitre du mauvais hasard, mais pas toujours. En effet nous avons vu que les chaînes A2 et B présentaient un biais d'alternance, et le graphique n'a pas aidé à repérer ce biais.

Enfin, il aurait été intéressant de proposer d'autres méthodes de visualisation des chaînes (par exemple celle décrite en 3.1.2, qui crée des images en niveau de gris) pour aider à repérer du mauvais hasard. Mais pour créer ces images en niveaux de gris, nous aurions eu besoin de chaînes beaucoup plus longues : nous n'avions pas les moyens de faire remplir autant de fiches.

CONCLUSION

Les scientifiques considèrent aujourd'hui que le hasard n'existe qu'au niveau atomique. Dès qu'on a assez d'atomes (cellules, pièces de monnaie), la loi des grands nombres fait que les systèmes sont déterministes, même si les erreurs de mesure peuvent rendre les résultats imprévisibles.

Dans la première partie de ce travail, nous avons présenté les travaux de Jean-Jacques Kupiec, dont la théorie expose de nouvelles sources de hasard au sein des cellules (la différenciation cellulaire comporterait une part de hasard) ou des molécules (l'emboîtement des protéines se ferait au hasard). Si on considère que le vrai hasard n'existe qu'au niveau quantique, le hasard dont parle Kupiec sera plutôt de l'imprévisibilité.

Dans la seconde partie, nous nous sommes intéressés au hasard produit par les ordinateurs. Dans la plupart des cas, ce hasard est construit par des algorithmes. Or un ordinateur est entièrement déterministe et, si on connait le programme, le résultat est entièrement prévisible. Ce qui est produit par ces algorithmes ne peut donc pas être appelé du hasard, mais doit y ressembler pour assurer la sécurité de nos comptes bancaires. Nous nous sommes donc intéressés aux définitions mathématiques de la qualité du hasard (complexité de Kolmogorov), puis à des tests statistiques moins efficaces mais qui ont l'avantage d'être calculables. Enfin, nous avons recherché comment l'ordinateur produisait du « hasard » : par des algorithmes, déterministes et prévisibles, comme celui de Von Neumann, ou par des appels à des sources extérieures de hasard (comme le bruit atmosphérique, déterministe mais imprévisible, ou des expériences en physique quantique, supposées être parfaitement aléatoires).

Enfin, dans la troisième partie, nous avons cherché à savoir si l'être humain est capable de produire et de reconnaitre du hasard. Pour la philosophe Patricia Churchland (Marchal, 2004), *« l'existence de tant de mouvements chaotiques avec les effets papillons correspondants est la raison réelle de la possibilité, et de l'existence, de la liberté : notre libre arbitre a constamment un grand nombre d'opportunités pour agir décisivement à un prix presque nul. »* Ainsi, pour cette philosophe, le libre-arbitre est lié à l'imprévisibilité. Ce libre-arbitre, cette faculté d'imprévisibilité du cerveau humain, permet-il produire du hasard de bonne qualité ? Nos expériences confirment les résultats relatés dans (Delahaye, 2004) : nos biais cognitifs (par exemple le biais d'alternance) font que nos productions de hasard ne respectent trop souvent pas les lois du hasard. Mais, alors que, comme dans (Delahaye, 2004), nous pouvons confirmer que le cerveau humain ne sait pas reconnaitre si une expérience, sous forme textuelle (binaire) est au hasard ou pas, nous avons pu confirmer qu'une représentation graphique de cette expérience améliorait fortement cette reconnaissance.

Nous avons été surpris au cours de nos recherches et de nos expériences par l'influence des biais cognitifs qui empêchent l'être humain d'évaluer correctement le hasard, et donc plus généralement, les évènements imprévisibles. De nombreux biais similaires ont été étudiés en psychologie cognitive, mais ils n'ont pas pu trouver leur place dans ce travail, focalisé sur la production de chaînes binaires. Par exemple la théorie du cygne noir (Wikipedia, 2008) explique que les êtres humains sous-évaluent la probabilité d'un évènement rare (qui a une faible probabilité d'apparition), même si ses conséquences peuvent être catastrophiques (par exemple l'explosion d'une bulle financière).

BIBLIOGRAPHIE

Barker Elaine et Bassham Lawrence Guide to the statistical tests [En ligne] // National Institute of Standards and Technology. - 2016. - https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software/Guide-to-the-Statistical-Tests.

Chevassus-au-Louis Nicolas Entretien avec Jean-Jacques Kupiec : « la probabilité a sa place dans le fonctionnement des cellules » [Revue] // La Recherche. - 2017. - HS 21.

Delahaye Jean-Paul et Gauvrit Nicolas Le hasard géométrique n'existe pas [Revue] // Pour la Science. - mars 2006. - 341.

Delahaye Jean-Paul Les dés pipés du cerveau [Revue] // Pour la Science. - décembre 2004. - 326.

Gillespie Colin RANDU: The case of the bad RNG [En ligne] // Why?. - 2016. - https://csgillespie.wordpress.com/2016/02/16/randu-the-case-of-the-bad-rng/.

Inconnu Le générateur de nombres aléatoires de la TI82-83 [En ligne] // MathémaTICE. - 2015. - http://revue.sesamath.net/IMG/pdf/activite_nbaleat_spemath.pdf.

Inconnu Visualise radomness [En ligne] // Type OCAML. - 2015. - http://typeocaml.com/2015/11/22/visualise_random/.

Kupiec Jean-Jacques L'ADN entre hasard et contraintes [Revue]. - 2013. - 81.

Kupiec Jean-Jacques Le Darwinisme cellulaire [Article] // Pour la Science. - 2009. - 63.

Marchal Christian Déterminisme, hasard, chaos, liberté. Henri Poincaré et la révolution des idées scientifiques au vingtième siècle [Conférence]. - Paris : [s.n.], 2004.

PHP mt_rand [En ligne] // PHP. - 2005. - http://php.net/manual/fr/function.mt-rand.php.

Pollock Rich Randomness in Excel [En ligne] // Bit Wrangling. - 2014. - http://blog.richpollock.com/2014/08/randomness-in-excel/.

SA ID Quantique Random number generation white paper - What is the Q in QRNG? [En ligne] // ID Quantique. - 2017. - https://marketing.idquantique.com/acton/attachment/11868/f-0226/1/-/-/-/-/What%20is%20the%20Q%20in%20QRNG_White%20Paper.pdf.

Wikipedia Complexité de Kolmogorov [En ligne] // Wikipedia. - 2005. - https://fr.wikipedia.org/wiki/Complexité_de_Kolmogorov.

Wikipedia Générateur congruentiel linéaire [En ligne] // Wikipedia. - 2005. - https://fr.wikipedia.org/wiki/Générateur_congruentiel_linéaire.

Wikipedia Générateur de nombres pseudo-aléatoires [En ligne] // Wikipedia. - 2005. - https://fr.wikipedia.org/wiki/Générateur_de_nombres_pseudo-aléatoires.

Wikipedia Middle-square method [En ligne] // Wikipedia (en). - 2004. - https://en.wikipedia.org/wiki/Middle-square_method.

Wikipedia Suite Aléatoire [En ligne] // Wikipedia. - 2009. - https://fr.wikipedia.org/wiki/Suite_aléatoire.

Wikipedia Théorie du cygne noir [En ligne] // Wikipedia. - 2008. - https://fr.wikipedia.org/wiki/Théorie_du_cygne_noir.

ANNEXES

ANNEXE A : A PROPOS DES GENERATEURS PSEUDO-ALEATOIRES

Nous avons trouvé sur un forum PHP un programme (écrit en 2015 par un certain *Phan*) qui permettait de générer des chaînes de bits avec différents générateurs. Nous n'avons besoin que de la fonction

```
get_random_string ($length, $function, $chunks)
```

qui crée une chaîne de bits de longueur `$length`, et crée `$chunks` bits à chaque fois en appelant le générateur `$function`. Par exemple, `$a = get_random_string(8, 'mersenne', 2);` va retourner une chaîne aléatoire de taille 8 bits. Chaque appel au générateur 'mersenne' (Mersenne Twister) va créer deux bits, il faudra donc 4 appels pour créer la chaîne. Ce programme peut utiliser plusieurs générateurs : Mersenne Twister, plusieurs générateurs congruentiels linéaires (celui de PHP, celui de Borland C++, celui recommandé par Donald Knuth, et le très mauvais RANDU), et l'algorithme de Von Neumann dont nous avons parlé dans le TPE. Comment utiliser cette fonction ? Pour créer 64 bits, faut-il appeler 1 fois le générateur en créant 64 bits d'un coup, 64 fois le générateur en créant 1 bit à chaque fois, ou entre les deux (4 appels avec 16 bits à chaque fois) ?

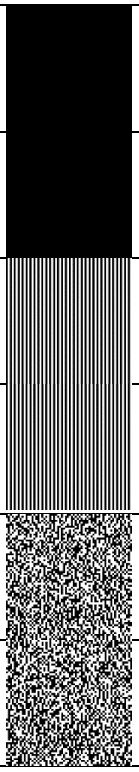
	Nombre de bits produits à chaque appel du générateur ($chunks)					
Générateur	*1*	*4*	*8*	*16*	*32*	*64*
GCL RANDU						Erreur
Von Neumann						Erreur
GCL Knuth						Erreur
GCL Borland C++						Erreur
GCL PHP						
Mersenne Twister						

Tableau 4 Représentation graphique des appels aux générateurs aléatoires avec différents paramètres $chunks

Nous avons donné dans le tableau ci-dessus les représentations de chaînes de 4096 bits (`$length` = 4096) créées avec plusieurs générateurs et différents paramètres de `$chunks`. Il montre que la valeur de `$chunks` change la qualité du hasard. Avec `$chunks` = 1, RANDU et Von Neumann ne produisent plus que des '1' (image noire). A partir de `$chunks` = 32 bits, le GCL de PHP et Mersenne Twister ne produisent plus que des '0' (images blanches). La valeur 16 nous semble produire la colonne

la plus équilibrée en qualité de hasard (même si le GCL PHP commence à avoir des problèmes avec cette valeur). Nous avons donc choisi $chunks = 16 pour le reste de nos expériences.

ANNEXE B : TESTS STATISTIQUES

Nous appelons expérience la création d'une chaîne élémentaire de 64 bits. Lorsqu'on utilise un générateur aléatoire, l'expérience est constituée de 4 appels au générateur, chaque fois pour générer 16 bits. En répétant N fois cette expérience, nous obtenons une chaîne globale de taille 64*N. Le programme que nos camarades ont pu écrire génère différentes séries statistiques. Quand nous les présentons, les x_i indiquent les valeurs, les n_i les effectifs et les f_i les fréquences (en pourcentage).

Calcul de la k-uniformité Ces tests sont réalisés sur la chaîne globale. On regarde les blocs de taille k, et les valeurs de la série statistique sont les blocs qui apparaissent dans la chaîne. Par exemple, avec la chaîne globale de taille 6 '000011', on trouverait 4 blocs de taille 3 : on lit '000' dans le premier bloc '000011', encore '000' dans le deuxième bloc '000011', '001' dans le troisième bloc '000011' et enfin '011' dans le dernier bloc '000011'. Ces quatres lectures ('000' deux fois, '001' et '011') sont les valeurs de la série statistique. Pour nos expériences, nous avons fait varier k de 1 à 3. Quand k = 1, notre algorithme compte juste les '0' et les '1' et c'est exactement le *frequency (monobit) test* du NIST. Quand k est supérieur à 1, c'est le test for *frequency within a block* du NIST.

Calcul des distributions Pour chacune des N expériences, on compte une valeur qui est soit le nombre de '0' dans la chaîne élémentaire (quand on calcule la distribution des zéros), soit le nombre d'alternances (c'est-à-dire le nombre de '01' + le nombre de '10') dans la chaîne élémentaire (quand on calcule la distribution des alternances

ANNEXE C : MERSENNE 20000

Pour évaluer les distributions des '0' et des alternances de nos deux expériences A (3.3.1) et B (3.3.2), nous avons voulu comparer avec les probabilités théoriques. Comme nous ne savions pas les calculer, nous nous sommes dit qu'on pouvait les approximer avec de très nombreux appels à l'expérience. Nous avons souhaité en faire le plus possible, et nous avons fini par en faire 20000 (à partir de 30000 nous avions une erreur de mémoire). Nous avons donc utilisé le meilleur générateur disponible (Mersenne Twister), avons généré une chaîne de taille 64x20000 = 1280000, et utilisé les programmes écrits par nos camarades étudiants pour faire les analyses statistiques. Nous présentons ici les résultats statistiques de l'analyse de la chaîne globale `$mersenne20000` créée par l'appel à la fonction :

```
$mersenne20000 = get_random_string (1280000, 'mersenne', 16);
```

(seulement partie de l'image)

TESTS DE K-UNIFORMITE

Test de la moyenne

x_i	*'0'*	*'1'*
n_i	640768	639232
f_i (%)	50,1	49,9

Test de 2-uniformité

x_i	*'00'*	*'01'*	*'10'*	*'11'*
n_i	320959	319808	319808	319424
f_i (%)	25,1	25	25	25

Test de 3-uniformité

x_i	*'000'*	*'001'*	*'010'*	*'011'*	*'100'*	*'101'*	*'110'*	*'111'*
n_i	160905	160053	160375	159433	160054	159754	159433	159991
f_i %	12,6	12,5	12,5	12,5	12,5	12,5	12,5	12,5

TESTS DE DISTRIBUTION

Distribution des zeros

x_i	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*
n_i	6	14	16	50	97	158	259	408	622	870	1275	1543	1783	1887	1968	1899
f_i %	0,0	0,1	0,1	0,3	0,5	0,8	1,3	2,0	3,1	4,4	6,4	7,7	8,9	9,4	9,8	9,5
x_i	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*	
n_i	1745	1538	1195	921	633	436	316	185	88	45	21	11	7	3	1	
f_i %	8,7	7,7	6,0	4,6	3,2	2,2	1,6	0,9	0,4	0,2	0,1	0,1	0,0	0,0	0,0	

Distribution des alternances

x_i	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
n_i	2	2	8	4	32	60	134	180	354	570	797	1103	1348	1586	1858	1937	2022
f_i %	0,0	0,0	0,0	0,0	0,2	0,3	0,7	0,9	1,8	2,9	4,0	5,5	6,7	7,9	9,3	9,7	10,1
x_i	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*	
n_i	1855	1624	1344	1089	799	561	326	190	110	60	24	12	2	5	1	1	
f_i %	9,3	8,1	6,7	5,4	4,0	2,8	1,6	1,0	0,6	0,3	0,1	0,1	0,0	0,0	0,0	0,0	

Annexe D : Mersenne 64

Cette chaîne a été créée pour le test de reconnaissance. C'est la même que dans l'annexe E, mais avec seulement 64 expériences. `$mersenne64 = get_random_string (4096, 'mersenne', 16);`

Tests de k-uniformite

Test de la moyenne

x_i	'0'	'1'
n_i	2019	2077
f_i (%)	49,3	50,7

Test de 2-uniformité

x_i	'00'	'01'	'10'	'11'
n_i	1007	1011	1012	1065
f_i (%)	24,6	24,7	24,7	26

Test de 3-uniformité

x_i	'000'	'001'	'010'	'011'	'100'	'101'	'110'	'111'
n_i	510	497	507	504	497	514	504	561
f_i %	12,5	12,1	12,4	12,3	12,1	12,6	12,3	13,7

Tests de distribution

Distribution des zeros

x_i	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*
n_i					2	2	1	3	1	3	3	6	3	2	5	8
f_i %					3,1	3,1	1,6	4,7	1,6	4,7	4,7	9,4	4,7	3,1	7,8	12,5
x_i	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*	49
n_i	5	9	5	4	2											
f_i %	7,8	14,1	7,8	6,3	3,1											

Distribution des alternances

x_i	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
n_i									2	2	3	4	9	6	3	8	4
f_i %									3,1	3,1	4,7	6,3	14,1	9,4	4,7	12,5	6,3
x_i	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*	
n_i	4	6	5	2	2	1	1	1	1								
f_i %	6,3	9,4	7,8	3,1	3,1	1,6	1,6	1,6	1,6								

Chaine obtenue

```
1001011100100001011010101011010000000100011110111001110100111 0001
0110101101000101100100111001001111101000000111110110011010100001
1101010000101110001110110110000011001111010000001101000101010100
1100101101110010001100111111110111010100010000011100111100011001
1010010011110001001111010100110100100111101010100000010010001000
0101100111011110010000000111011001001001010100110000100010100100
0100001101100100111001110110001110100011001010001110001000001010
1101101000111111101100101101111110000111001101011000111011110111
1001101000001101010111111000010111011111010001000000100110010100
0100010100101001110011111001110001010101000101011000111111011111
0001111000001010001010100011110111011101110101111100110111011101
1011111111111011111011101110010011011011100110100101010100001011
0000111011111100101010000111010000100000010110101111010001010001
0011101000000000011001010010000011001010101110110010001110000111
1110011000000101110111011111010000111011010111100101101010101 1001
1111001000000100010100010100001100110001011101111011101100110011
0000110010111101110100000111011011101011001000101111011111010111
0101001111000011001101010001100011101100100011000001100010000101
0011010010111001111001110000110110001100010101010001111111100111
0001001011011110000111011010001011100001010010000111110011100000
0000101011000101111001001011000101100111011100010110101000100001
1111100000100101001100000010011101000011001011101110110101110001
0010001000000000110110010000101111010101110001100110111110111001
0110011010001011101110001111000010101101011101010000101101010111
1111110110111000111011011100110110000111101110011100111001010011
1000000111111001001111001001000111111010001110101001001100110001
0110011001111001001100000000111100110001101111100100010010011110 0
1110011010111000100100011111011111111110111010111101100000000 1111
0001011001101010010010111111000010111001101001011010111011100111
0010000111111100001100110000111011011101011000000101001100111111 1
1101010100100111011001100110010011100101101001000010001001100010
1100100011100101001000110111001000011110001111100000000100101101

1111001110010001110011111101111010111111111001101011111011011 0000
0010111001010100110100011101000000101010001110000101000011011011
0001100011101100100010100000011011011011000000101111101111101111
1000011100100001111111000010001011011001101100100100000101110001
0010100101001001111000000000000011000001010000111110100110011111 0
1110101000000100101101110110110101100010010001011101100001110011
1001000111100001000101111111101010110100000101001011101010000101
0010001010001100111010100101111001101111010110100001001010011100 0
0001100101100010001000101110000111011111110011111111110100010000
1011000110100001111100000000110100011001000000111000000101111101 01
0011001011111000011111111101111001111000011100101010101010110111
0111101111010011100010100011010010110010110001111111110111000111 1
0000001010001010111010111000100010110101101001011001010010111101
1011001101100110100000111011010101000101101011010011101010110111
0011000110011010011010000011010001011010110111100001110010011100
0110100111111101110000011011101000110111010111100110011001001100
1001011110011101100100111101000110100101110010111110000110101010
1110110111101111100001111010101011110100101010100011101110001011
1100110110011010000000101100001111001000110010101110000100010100 0
1001010011010001010111000000000110100111100011011000111010001011 0
1101001001111000010000000100111010001001001100011000010011110111 1
1011111010000010001010100101000101101110010101110011010000011011
0101011111001010001110011101111111110011000111100011101000001001
1111100000010111010101001110110001101001000100010110111111010000
0101000001110110111111111111010101111111111011100110001010011011 1
0000000111001011100010001010100001111110110000010010011101011000
0000101110011101000001100100100000000100101001011110000011111011
1111011010110111110100011010101011101011111110000010110000110000
0011101010101001111010010101010100100010100010001010011110100111
1110111101101000111000110011000100000001011011011000000101010101 1
1110101010101010011101011011000110110100001100100100100011111011
1101100011011000101101111000000111110100100010111110000110010001 0
```

ANNEXE E : QUANTIS 64

Cette chaîne a été créée pour le test de reconnaissance, et construite avec des chaînes élémentaires de taille 64 générées par une carte quantique sur le site http://www.randomnumbers.info/.

TESTS DE K-UNIFORMITE

Test de la moyenne

x_i	'0'	'1'
n_i	1971	2125
f_i (%)	48,1	51,9

Test de 2-uniformité

x_i	'00'	'01'	'10'	'11'
n_i	903	1067	1067	1058
f_i (%)	22,1	26,1	26,1	25,8

Test de 3-uniformité

x_i	'000'	'001'	'010'	'011'	'100'	'101'	'110'	'111'
n_i	416	486	524	543	486	581	543	515
f_i %	10,2	11,9	12,8	13,3	11,9	14,2	13,3	12,6

TESTS DE DISTRIBUTION

Distribution des zeros

x_i	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*
n_i					1	1	1	2	4	3	4	9	7	7	4	3
f_i %					1,6	1,6	1,6	3,1	6,3	4,7	6,3	14,1	10,9	10,9	6,3	4,7
x_i	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*	49
n_i	8	2	4	3	0	0	0	0	0	1						
f_i %	12,5	3,1	6,3	4,7	0	0	0	0	0	1,6						

Distribution des alternances

x_i	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
n_i								1	1	2	0	3	4	4	4	3	5
f_i %								1,6	1,6	3,1	0	4,7	6,3	6,3	6,3	4,7	7,8
x_i	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*	
n_i	4	7	7	6	7	3	0	2	0	0	1						
f_i %	6,3	10,9	10,9	9,4	10,9	4,7	0	3,1	0	0	1,6						

CHAINE OBTENUE

```
0010101001001001100110111001010011110010111001010111101010011111
1011100001011011011111001100101101011010000000010101100100101010
1100111111011010100011010011101111011111000000000001001011111000
0011011000010111001000100010011000101001110101110010110010100111
0010100010101011011110011110011101111011010101011110111001001001
1110110100110111101000001111111100000100010001001111010101010111
0001111110000110000000010101011001101011101001011100101111101110
0011000110111000100110011011010001001100110110010010101000111001
1101001101110011110101111101100100000011011110101101111010010101
0110010110111000010010101111110110101001101010110100011101000010
0001001100011100111001010100000101000101011100000000000000001000010
0110101101011101011110000000111110101001010101010101110010110100
0111000000100100110110011011011110101110111010100100101111010011 0
0101001110010010100110010101001110101101011110101010110010110011
1010110101011001000010101100100110010001100001100110010010001110
0001100100001110101010101001000111110011100001001110000101110000 00
0100101011001100000111000101010110111010000011001011000110011000
1010010110101110111100101100011111100001001011010001010111101110
0100101111000000101111011101111101000000100111111001111000001111 1
1011101011001110101100111011101101001101000101011010010010001001
0000111011011000101110000000101111010011001000000010110110010001 11
0101010100110000111000100101110101111110010010110010110000101000
1100100101110101111110001100101010010011110111111001010011011001
1001010101101010001110000011111001110111101101011101101010100011
1000000100010110001100110011011111010001110101100001100011001101
1010111100100101111011101001001010100000111101110111101110100000
0010010100101001011111010010100001110101001101001000111011100010
1111011001100000110011011100011011010110101010011111011111111010
0011101011110011011111101010011001111011110001010110000000101010 0
1101111000111110011101000101010101010110111010000111101010001101
1010000010100111011000100110111001111010110011001011011010110100
0101000101000100010101111001001110101000010011010100111100000111
1000100100110100000001011011111011001100000111001000100110000011 0
0010010010000101000101010011010110100111100000111101001100010011
0000110000110011011011100100010111010111000011010100000000010101 1
1011101010111111101110110100101100001100111010100110110101001110
0001101011110110001010111110110010110111011100101001001111001111
0101000011000011010111100100001110110011010011100011111100000111
0100111111001010101111110011010111101111100011010001100010101100 1
0100011110010011010110010111110001100010010010110010110101111100
1100110001011011011001101101011111001011000001111001111000000101
0100011010100010011011100101000111100011000101111011111000001101 1
1010000101010011011100010101000001101100110111101100100011110011
0010110010011101100110011011111101100000100110101101000111111010
0010100010111111011110101101111111100101001010111101100110110101
0110001011100001000010111010101100110101110110101100011010111100
0010101110101111110101111001101110110011100111001101011000111100
1110001000000101001100001000110110110101110011000111110110111100
1110111111100100000101001010111101001110000011000000110000110101
1011000011111111111100111001000101101011100010011010010110011000
1000010110101010110001100001101110111001101011001011011100000111
0100011101010101110011101110011010011010111110101001101100010000
0111101000101001001110001011011001010111000011101101100001000101
0001011101101001100001011101100100110010110000100101010010110110
1000010011001011011101100110100101111111110100010010010110000100
1001100011000111101101111111011000010011000101111111111010101100
1111001101101011101000110011011000001001011010011010111001101010
0001110100100001110000101110100000000111000010100111001111010100 0
0000101110000011101111110011011000111111111011000100011010011000
0110101111010100110111101101000110111010011111101001100001001000
0011111011100001010111100000011010100000001011010110100110011001 0
1111011011101101001001100111111111100111011101001000001101111110
0111111101101010010000110010111111111110111111011101100000111011 1
0100110110010100011000100111100000100100110011001101101001111100
```

ANNEXE F : RANDU 64

Cette chaîne a été créée pour le test de reconnaissance. C'est la même que dans l'annexe F, mais avec le générateur RANDU. `$randu64 = get_random_string (4096, 'randu', 16);`

TESTS DE K-UNIFORMITE

Test de la moyenne

x_i	'0'	'1'
n_i	2049	2047
f_i (%)	50	50

Test de 2-uniformité

x_i	'00'	'01'	'10'	'11'
n_i	1027	1022	1021	1025
f_i (%)	25,1	25	24,9	25

Test de 3-uniformité

x_i	'000'	'001'	'010'	'011'	'100'	'101'	'110'	'111'
n_i	479	548	472	550	549	474	549	475
f_i %	11,7	13,4	11,5	13,4	13,4	11,6	13,4	11,6

TESTS DE DISTRIBUTION

Distribution des zeros

x_i	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*
n_i						1	2	2	0	1	4	3	5	8	11	6
f_i %						1,6	3,1	3,1	0	1,6	6,3	4,7	7,8	12,5	17,2	9,4
x_i	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*	49
n_i	8	3	3	2	3	1	0	0	1							
f_i %	12,5	4,7	4,7	3,1	4,7	1,6	0	0	1,6							

Distribution des alternances

x_i	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
n_i										1	2	0	3	6	10	12	8
f_i %										1,6	3,1	0	4,7	9,4	15,6	18,8	12,5
x_i	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*	
n_i	12	5	2	1	2												
f_i %	18,8	7,8	3,1	1,6	3,1												

CHAINE OBTENUE

0001010010011001001111011100101110111001011000010010110000100011
1000010001101001100011010011101110100111101100011111011100010011
1110010100111001101011111010101100001111000000001001011010000011
1000011100001001100101010001101110111111010100010011110111110011
1011100111011001001011011000101110001000101000011001100111100011
1100110110101001011010001111101100111010111100011011000011010011
0001001001111001001101110110101110100110010000011111001011000011
1101100001001001100010001101101110011010100100011100111110110011
0110111100011001010011010100101111100111111000011011011110100011
0010011011101001011101001011101101011110001100010001101010010011
0100111110111001111011110010101111001101100000010110100010000011
0011100110001001101011001001101100000101110100010001000101110011
0011010001011001100111010000101111010111001000011000010101100011
1001000000101001101100000111101100010001011100010011010001010011
1001110011111001110101101110101110000100110000011000111001000011
1010101011001001000000000101101100000000100010001000000110011001 1
0000100110011001000111001100101101010110011000010000000110010001 1
0000100101101001000111000011101101010100101100011111111000010011
1111101000111001111011101010101111001100000000010110010000000011
0010110000000100110000100000110111000110001010001101001001111001 1
1110111011011001110011001000101101100101101000010011000011100011
1001001010101001101101111111101100100111111100010111011111010011
0110011101111001001101100110101110100011010000011110100111000011
1011110101001001001101111101101110100111100100011111011010110011
1110010000011001101011000100101100000100111000010000111010100011
0010101111101001100000111011101110001011001100011010000110010011
1110010010111001101011100010101100001010100000001000111111000011
0101111010001001000110111001101101010010110100011111100001110011
1110100101011001101111000000101100110100001000011001110001100011
1101010100101001011111110111101101111110011100010111101101010011
0111000111111001010101011110101100000000111000000100000101010000 11
0000111111001001001011110101101110001110000100011010101000110011

1111111010011001111110111100101111110011011000011101101000100011
1000111001101001101010110011101100000001101100010000010100010011
0000111100111001001011011010101110001001000000011001101100000011
1101000100001001011100110001101101011001010100010000101111110011
0010001111011001011010111000101101000010101000011100011111100011
0101011110101001000001101111101100010100111100010011111011010011
1011110001111001001101010110101110100000010000011110000011000011
1010001001001001111001101101101110110100100100010001110110110011
0101100100011001000010110100101100100001111000010110010110100011
0011000011101001100100101011101110111000001100010010100010010011
0111100110111001011011010010101101000111100000011101011010000011
10000011100010011000101010011011100111111101000111011111101110011
1001111001011001110110110000101110010001001000011011001101100011
0001101000101001010011100111101111101011011100011100001001010011
010001101111100111010100111010110111111101100000101111110001000011
0111010011001001010111100101101100011011000100010101000100110011
1111001110011001110110101100101110010000011000011011000100100011
0001001101101001001110100011101110101110101100010000110000010011
0010010000111001011011001010101101000110000000011101001000000011
0111011000001001011000100001101100100110010100010111001011110011
0101100011011001000010101000101100011111101000010101111011100011
00011100101010010101010111111011000000001111100010000001011101 0011
0001000101111001001101000110101110011101010000011101011111000011
1000011101001001100101011101101111000001100100010100010010110011
1100111000011001011010100100101100111110111000011011110010100011
0011010111101001101000011011101111100101001100011010111110010011
0000111010111001001011000010101110000100100000001100011011000011
1010100010001001111110011001101111101100110100011100011001110011
0101001101011001111110100000101111101110001000011100101001100011
0101111100101001000111010111101101011000011100010000100101010011
0001101111111001010100111110101111111011110000011111001101000011
1101100111001001100011010101101110101000000010001111110000011 0011

ANNEXE G : EXPERIENCE A (92)

Cette chaîne a été créée par sondage pour l'experience A. Elle contient 92 expériences. Pour le test de reconnaissance, nous l'avons coupée en 2 parties (64 premières et 64 dernières expériences).

TESTS DE K-UNIFORMITE

Test de la moyenne

x_i	'0'	'1'
n_i	2896	2992
f_i (%)	49,2	50,8

Test de 2-uniformité

x_i	'00'	'01'	'10'	'11'
n_i	1347	1549	1548	1443
f_i (%)	22,9	26,3	26,3	24,5

Test de 3-uniformité

x_i	'000'	'001'	'010'	'011'	'100'	'101'	'110'	'111'
n_i	604	743	794	755	743	805	754	688
f_i %	10,3	12,6	13,5	12,8	12,6	13,7	12,8	11,7

TESTS DE DISTRIBUTION

Distribution des zeros

x_i	14	16	24	26	27	28	29	30	31	32	33	34	35	36	37	38
n_i	1	1	1	4	4	10	8	7	11	14	6	5	9	4	1	2
f_i %	1,1	1,1	1,1	4,3	4,3	10,9	8,7	7,6	12,0	15,2	6,5	5,4	9,8	4,3	1,1	2,1
x_i	40	42	46	51												
n_i	1	1	1	1												
f_i %	1,1	1,1	1,1	1,1												

Distribution des alternances

x_i	16	19	20	21	24	25	26	27	28	29	30	31	32	33	34	35	36
n_i	1	2	2	1	2	1	5	1	4	1	5	9	9	7	5	3	8
f_i %	1,1	2,2	2,2	1,1	2,2	1,1	5,4	1,1	4,3	1,1	5,4	9,8	9,8	7,6	5,4	3,3	8,7
x_i	37	38	39	40	41	42	43	44	45	55							
n_i	3	5	7	1	2	2	2	1	2	1							
f_i %	3,3	5,4	7,6	1,1	2,2	2,2	2,2	1,1	2,2	1,1							

CHAINE OBTENUE

0100011011100101101110010110111001001100100101000110100101011101
0110001110010011110001001111000110001000100111001111000011100110
0101011000000110100111010101010110000000000000010000000000000001
0100010100010111011011011001001101110100111100100110111001010010
1011000111011011100001101101111010100110011010110111100011011000
1001100101100010101101101111001010000010101001001010010101001011
0000000010000111111001010010010100111110010101010010010010011111
1010000011111111000101010101011101010111001100011101010101111001
1011000110010011010110010010010100101110110100101001010110011111
1111111011111110001100111000000100001111000111111111110101010011
0111000100100100000111100001011001000110011111110000010010001100
0011001010011010010011110111001000001110101101110010110110101110
1010011010110001010100011010010101111001001110001100110100110010
0111010001011001111001011010101110011111001100101001110111010000
1001110100011110101100011111011010100110011001011100101111111100
0100110111000110000101011011001001101001101010010101000111001101
0110001101001110110110011000011001011010100110001101101100010110
1001111010010110001011000000110101011000100001010110010101110001
1000101110110100011110101010011001011101010110011100010101100111
0110010100011001001111010101100100110101110000101011001000101111
0001010111000100110101010010000011001111000110001111000011110000
1101110010011010100000110100110111101010001001101110111110011011
1101101110110110111000111100010001101001110101110010010111100111
1100011011100101100010110100111011011111110001011000110011101101
1010111010010111001010111001010111010000101010111000101000101010
0001011010100001010101110111100010011100110011111111001100110010
1000110010110011110011010010110111101011100010011001110011101101
0100100000010000000000000000000010011011111101011111111111010000
0010111010101101001010010010111001001110011101000110110100100000
0100100010000100000100000010000001000010000100001000010000100001 0
0101101100110111011000111100001111101110000111001110110111000110
1100010111100101000001011010011100111010000110101110101001011000
0111001010011110100110111100000110111010001101110001011100001101
0010011101001111010001011010100110011101010001110101001101100011
1001100011100011110110101110001101110110110111101000011101011110
1000110001010100011110110110101010101110000000111101011001000000
1111011101111111110111111111110100001110101011101110001111111110
1001101001101001010100110010010011010010100101010101100110010110
0110011101100001101001101110000011010011001011001011000011100111
1011110010101011100011000110101010010101010111110000010101010010
1001111010010110001011000000110101011000100001010110010101110001
1110011111100111111011111011100011111100011111110111111011111111
0010110011010000101100101010001111000111010110010100000011111000
0000001110000110011111111101011100000111111001110000111011011111
0000101111001100101010001111111001111100010101010001010001100111
0010000111000011111000110001110111110000111111001110001111000111

0101011110101001110111000110101011111000111100011000110011100011
1000100011100111000110011100010101110001001110001100011100011000
0101100100100111000111001100110001101001100111111100011010111000
0100101010101000101010101010101001010100101010010100100101010101
0110011100111101011110100101001110101111010001110100110011110000
0100111001011010001101010100000110011111000101010100010000010100
1010011010111000110000111001100010111001001101100100100010111000
0011010111001010111000111010101110001110101011101110001010111010
1011001110011000010110111100011101010111001110011110000110001110
1000110011101010011110011011000110100110011000111100110011000011
1110001110101000011100101010101000011111100011111000101010101010
0001010011110011010010111011000110011110100111010000011101000111
1110011001001111001010001101001100011001001110011010011110010010
1001111100010101100000011111111101010111010100010101111010111000
0110000101001110101011000110101110110010001101001100010101100011
0111101011100001010001001010101111011001001100101011101001001001
0100011110001111100000111010011001111111100011010101011100000000
0100111001111100100110001111010010100110000110010001010011001101
1110001010000011001001000000111010010000010001010111000100100101
0000010100001111100110100110110000110101001110101110110100000010
0101110110000010000011101110010111100011100000001111100011101000
1001011101100110001101000101010010110110011100110011001101010010
1100111000110000001110011101110011100011000110001101101101011111
0111010100111001010110100110101101011100110101100111011010110101
0110011100001011001011010001110111100100001011001000010111011011
0000100100011100111001011001001101110001100110010010001110011100
0010110111001110000100101011100010100111100011010101100011001111
1010011000010111100100011101010100001111100010101010001101101011
1011011001110011000111000111100011011111110001110111100101010011
1011110000011010000111111001011110001010110111011101011100001101
1011001001110110101100110010000101011101101101011110000111010011
1100111100001100110001110111000001000111110101010111000111000111
1010111110110111010111000001100010101010111111000000000111100011
1001010001111001010111000010101100000111010100000100111100010011
0110001100100011100101010010100110001110001110011010000100101001
1101100101101000110101110001010101000000101101010100011010101011
0101110101000101001110010110010101110101011010001110010010101010
1010011100111110001110001100101010100100100101010101001110011010
0100001001111000111010100011000011100010111000000101001110001010
1010011011011010101101010101000110110100110011010101010011100110
0001100111001101000100011101001011100001101011011111011101101010
1011001110101101100011010111010110110100111001010011101001100100
0011101011011011011010101011000000101011110101011101011010100111
1110100110000110000111010101010001110011000011111101010101011111
0011010010100100001110010010010000100100001000100011100010100010
1100010110100010100011011111010100010110011100010010110101101011

ANNEXE H : EXPERIENCE B (64)

Cette chaîne a été créée par sondage pour l'experience B. Elle contient 64 expériences.

TESTS DE K-UNIFORMITE

Test de la moyenne

x_i	*'0'*	*'1'*
n_i	1999	2097
f_i (%)	48,8	51,2

Test de 2-uniformité

x_i	*'00'*	*'01'*	*'10'*	*'11'*
n_i	859	1140	1140	956
f_i (%)	21,0	27,8	27,8	23,3

Test de 3-uniformité

x_i	*'000'*	*'001'*	*'010'*	*'011'*	*'100'*	*'101'*	*'110'*	*'111'*
n_i	370	489	625	514	489	651	514	442
f_i %	9,0	11,9	15,3	12,6	11,9	15,9	12,6	10,8

TESTS DE DISTRIBUTION

Distribution des zeros

x_i	*10*	*22*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*	*34*	*35*	*37*	*38*	*39*	*41*
n_i	1	1	2	1	4	1	7	6	9	13	7	2	1	3	4	1	1
f_i %	1,6	1,6	3,1	1,6	6,3	1,6	10,9	9,4	14,1	20,3	10,9	3,1	1,6	4,7	6,3	1,6	1,6

Distribution des alternances

x_i	*11*	*15*	*22*	*24*	*25*	*26*	*28*	*29*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*
n_i	1	1	1	1	1	1	4	2	3	1	3	1	11	1	9	5
f_i %	1,6	1,6	1,6	1,6	1,6	1,6	6,3	3,1	4,7	1,6	4,7	1,6	17,2	1,6	14,1	7,8
x_i	*39*	*40*	*41*	*42*	*45*	*49*	*52*									
n_i	3	7	2	2	1	2	1									
f_i %	4,7	10,9	3,1	3,1	1,6	3,1	1,6									

CHAINE OBTENUE

1010010100011010110110011010100001010011110100100010110000000000
0111111010111101110000110001100001011000110000111001100100011000
0101110101001000010110010110100100001010101100101010001000101100
0101100110111001100001101001101000111010110010101100110100110110
0010010011111011100010001101111111000000010101101101001001011001
0011001111110000010110011100011111101010111000100100001111111001010
1001011001001001000101111001110100111100100011010110101010100000
0110111010100001010100101110110101110010101011011101001100111100
0100101110010001101100000001011111010111011000100010101010101011
1010010101010011001111100100010111011010101011010001000011110111
1001000110010100001001000100100010010100011000101100100100101100
1001010101000111101110010011100010010010111011100100100101111000
0101101100111001101010110010110110101110101100100101110001101100
1101010100101011110111011011010101101110100101111111101010101011
0110010110101100010110010110110101100010100011010110110011001011
1010011000110101111101011000101111010010011011001000101010010100
0110010110100001101110010010111010111010000101110100100111010001
1011101010110101110000000101111100110000001101011101101111111000
0011001111000101100101101001001010001101110101010000001010100011
1010110001010001101100111011011001001000100011100000101100110110
0010011001001100000110101000111000110101101100101011100101100101
1010001101100110100000000111111100001110111011100000110101011001
1011000111001101001001011101101000100111001111000111101010000110
0111011000110111010000110110100001101010011000110110000011101101
0111100010111110011111001001110101000101111010011101111011011001
1110111100010010101010110101010110101010010101011010101001010110
1010011011011001110001100101001010111010110101100011101010011110
1011010110001111011100110111110001011001101010110110100110111000
1000110101110001101010010010111010100101010111000111001110001000
0011011100101101001001000110010100101010011110101010110100111001
1011001101101101110011101101010001111010110110011100111001011000
0110101010001101010010001010101110101001111111110000001010101010

0111010110010110101001010101101010101110010110101010101011110101
0011111111111010111011111101111111111111111111001111111110101111
0100001101101110011101001001011110011010100100101011110001110111
0100000110110111010010001101010101000100101001010111100010001000
0011000110100010011110001100001101000111011100001000000011110000
1100111011111111001001100000000111101011101101111000111110000011
0101011101010111101010001000101110010011101010101000110011110001
1101001001011001100010010111010110101110010100101001101001100101
1100011100011001110101101011010110100101010110001101101010100111
1010001101011101101001000011000001100110101111100110110101001110
1001010000111001010011001111001100011110100101010101011010101001
1010001101001110010001111001101001011001011010111110010100101101
1001011100010010110010111011001010011001101001100100011110010010
1110100001110100111100110010010010111011101001101010011001101100
0011101101110100101011100100110110101001100000011101110010000001
0101010110101010010101010101010101010101011110101000010101111010
0111001010000100110011110000100101010100101101111110010011010111
0000111011110001001011011100110000111000110010010110011000011011
0100100000110101011110001111000110101110110010011001110001010101
0101011010110000001001101000111001101111010110101101100010101000
1000000101000010011001000101100001011000011101000111111010000001
1010100001111000011011101011000000111111111001111010101010000111
0111101001001011011111101001011010100110110110011111111001010101
1010101011110000100101111100100011111111011110000000011101110000
0110100101000100010110101011000001000010011001011100100000011110
1011100001111100110100110010011011001010010001101001111010010110
1010010110100010100100000110100110101010010101010101000100100010
0000001111111111111111111111111100000001100000011111101001000111
0100111100101001011101000011111001001100011011100000111101010100
0110101110100001010111001010111101010001000011101110000011010111
0111000101011101111111000110001010110110111101100010101000011111
1010001001000111010001011001101010100101000010010011001100101001

ANNEXE I : CALCULATRICE TI 82 (64)

Cette chaîne a été créée pour le test de reconnaissance. Elle a été créée par des appels à la fonction *aleat* de notre calculatrice TI82.

TESTS DE K-UNIFORMITE

Test de la moyenne

x_i	'0'	'1'
n_i	2028	2068
f_i (%)	49,5	50,5

Test de 2-uniformité

x_i	'00'	'01'	'10'	'11'
n_i	1006	1022	1022	1045
f_i (%)	24,6	25	25	25,5

Test de 3-uniformité

x_i	'000'	'001'	'010'	'011'	'100'	'101'	'110'	'111'
n_i	489	517	509	512	517	505	512	533
f_i %	11,9	12,6	12,4	12,5	12,6	12,3	12,5	13,0

TESTS DE DISTRIBUTION

Distribution des zeros

x_i	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*
n_i					1	1	0	2	3	3	2	8	4	9	4	7
f_i %					1,6	1,6	0	3,1	4,7	4,7	3,1	12,5	6,3	14,1	6,3	10,9
x_i	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*	49
n_i	2	3	8	4	1	0	1	1								
f_i %	3,1	4,7	12,5	6,3	1,6	0	1,6	1,6								

Distribution des alternances

x_i	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
n_i									3	3	4	4	1	1	7	10	8
f_i %									4,7	4,7	6,3	6,3	1,6	1,6	10,9	15,6	12,5
x_i	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*	
n_i	3	3	9	3	2	0	0	1	1	1							
f_i %	4,7	4,7	14,1	4,7	3,1	0	0	1,6	1,6	1,6							

CHAINE OBTENUE

```
1001000011101100001111111101010111110111000011110100001111001000
0100000000111000100100101101000000000101111000110010111111100010 1
1110100111000101111100011000100000110100001011001111111110011100
1110000010011010011101000111001100100010110111110011100111110101
0110101011110011100100010010100101101111111001111010011111001111
1111101110100011101101010000001111001010110001111101011001010011
0001100110101100101111011010100100110101011101100010101111100000
1011011011001111101111101100000000110110100111110000100111110100
0100011001111111100010010111110101100001010011010000111001001000
1100001000010110011111011100101010011001110110101000101000000000
0101100110101001001010101000101101011101101111110100001110001010
1101011011010100110011100101010111100111101010111000100111110011
1000010010111001010010011001001100001101000000000100110000010011
0110111010010111011100101101000000111111001101011001111010100101
1000110111011110001010110101110101000110101010101010111010111101
1010001011101100101111010110010001101111110011001111101100010000
0101110011111001111000101011000100101001111000101001110110010101
1100011011111011011010001001111010111001110100010110010000100001
1010100111100011010111001001101010010001111000111101100100100100
1111011000000010100001101001101110100100111011011110100110100101
0111010000110011110000101111101000010110011010000001101011100100
1101001110000101110111000110000000010100100100101011100011000101
0100101111100110010011101001111111011000010101001100001001110111
1111010001001110010010111100001001010100010110001100001011100111
1101001000110111111001100110110010011000111001110100001101110110
1110000110110110111010001110010100001001000101000010101110001000
0100101000111001010100100001111001000000000000100000111111111110
1010000101101101110110110011000001001001000001101100101100110111
1010101011001111111110101110000111100111011110011101000001111100
0000000001011110001010010110111100011100111100000101001000000000
1110011001100000011001011010000000100101010011110110001111100101
0111100000100100000000000111110101010111000111101111000101110001 0

0011000101010000101111000001000111000010110011000101001111110110
1111001101100101110000010110001001011001011010000101110011100010
0110000101110010101011101100010000110110110101101011010101011001
0000111110111011111110000000110001100110010101100100011001010111 1
0001110000000110111011111111110000011010101011001110101001100000 00
0011001100101101111111100101000011111011111110000101110000100000 1
0111010010100110110100110001011001111001101011111111011111110111
1011100000100101100001000011100101001010010011101100111001001010
0011000111010100110101110110000100110110111001001000101000110111
0001110111011110101110110011101011101100010110000110001101101110
0001001010101011111111001111001100101001011010111111000010011010 1
0100100000100101000111011011010100111111111100001111001101101010 0
1001001001010101000010000101011010101000010000110101111000001110
1001000010011001000100110100110101110011101011011100101000001001
0011011111110010010110010110010111000000001000001000001111100110
0100111110101010000111111110000110010101110111001011111101001011
0110111000110010101110010111010010101110010011110110101011001111
1010110011001110011000100111100111000000010001011101000101010111
0100111001111010011111010101100110110011100111101110111011101111
1110011001011100001000110001000101110010110110111101100010111011
0001100001101100011101100010010110101011000100100100100011001101
1010010001111101100000110111011100001101001000110111001110100110
0101011100111001110100100110101100111100010110110001101000111101
0010111011111000110011011100100011000001001111011000110101011100
0011001111110000110100001010000000011110111111001000000101011010 0
0101010110010000000011000000111000011111010110100001001011011011
0011110011000101100010001010010110000011111000000101011110010000 1
1010010010110001110010111011111110001101100011010010010100101000
0010100010011111111110100110100001011110001001110001100110100000
1000100000001001110011010000111011110000000001111001101000010011 1
0010011111100110001011111101011001010101110010011110101001111011
1000010010001100100111000001001000011011000100111010110101101001
```