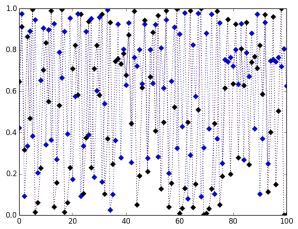
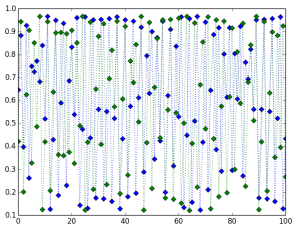
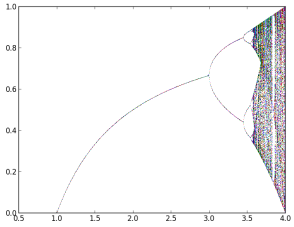
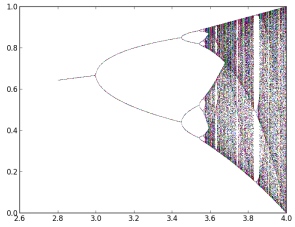
Stage Février 2016 Du 08/02/16 au 12/02/16

Cryptographie et chaos

Le but du TP est d'utiliser certaines propriétés permettant de coder de la suite logistique pour trouver une méthode de cryptographie : il s'agit de la grande sensibilité aux conditions initiales ainsi que le fait que toute région de l'intervalle $[x1, x2]$ est traversée une infinité de fois par la suite logistique pour un paramètre λ suffisamment grand, avec $x1$ et $x2$ deux éléments de $[0, 1]$ dépendant de λ. Dans les figures suivantes, on a tracé les 100 premiers termes de suites logistiques de paramètre $\lambda = 4$ dans le premier cas, et $\lambda = 3.67$ dans le second cas.

On voit que les intervalles $[x1, x2]$ semblent être $[0, 1]$ pour $\lambda = 4$ et $[0.11, 0.98]$ pour $\lambda = 3.87$. Valeurs que l'on retrouve dans le figuier de Feigenbaum.

Principe On choisit un paramètre λ 'proche' de 4, et on fixe l'intervalle $[x1, x2]$ correspondant à l'intervalle des valeurs accessibles par une suite logistique de premier terme autre que 0 ou 1. On découpe cet intervalle en 26 intervalles réguliers (correspondants aux 26 lettres de l'alphabet), mais on pourrait choisir de coder des messages en tenant compte également des chiffres, des espaces, voire des ponctuations ou des minuscules et majuscules... (ce que l'on pourra faire dans un second temps lorsque l'on aura fini la première version du codage...). On attribue à chacun de ces intervalles une lettre : 'A' pour le premier, 'B' pour le second ... 'Z' pour le dernier. On travaille selon le tableau suivant :

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Puis on choisit un nombre flottant entre 0 et 1 : u_0 ne correspondant pas à un point fixe de la fonction logistique ni à un antécédent de ce point. Ces deux nombres λ et u_0, ainsi que $[x1, x2]$, formeront ce que l'on appelle la clé de chiffrement.

Chiffrement.

Ensuite on considère le message à coder.

On itère la suite logistique avec ces paramètre et premier terme jusqu'à arriver dans l'intervalle correspondant à la première lettre du message à coder. On retient le nombre N_1 correspondant au rang du terme de la suite se trouvant dans l'intervalle correspondant à la première lettre du mot.

On poursuit les itérations de la suite jusqu'à arriver dans l'intervalle correspondant à la seconde lettre du message à coder : on retient le nombre N_2 tel que $N1 + N_2$ soit le premier rang après N_1 du terme de la suite $(u_n)_{n\in\mathbb{N}}$ se trouvant dans cet intervalle. Et on réitère le processus jusqu'à avoir traité toutes les lettres du message.

Exemple

On veut coder le mot 'HELLO' avec la clé : $\lambda = 4$, $u_0 = 0.6$, $x1 = 0.2$ et $x_2 = 0.8$.

Le premier terme tel que $u_n \in \left]0.2 + 7 * \dfrac{0.8 - 0.2}{26}, 0.2 + 8 * \dfrac{0.8 - 0.2}{26}\right[$ est $N_1 = 56$.

Après 56, le premier rang à partir duquel $u_n \in \left]0.2 + 4 * \dfrac{0.8 - 0.2}{26}, 0.2 + 5 * \dfrac{0.8 - 0.2}{26}\right[$ est $N_2 = 144$ donc on retient $N_2 - N_1 = 88$.

Après $N_2 = 144$, le premier rang à partir duquel $u_n \in \left]0.2 + 11 * \dfrac{0.8 - 0.2}{26}, 0.2 + 12 * \dfrac{0.8 - 0.2}{26}\right[$ est

$N_3 = 189$ donc on retient $N_3 - N_2 = 45$.
Après $N_3 = 189$, le premier rang à partir duquel $u_n \in \left]0.2 + 11 * \frac{0.8-0.2}{26}, 0.2 + 12 * \frac{0.8-0.2}{26}\right[$ est $N_4 = 246$ donc on retient $N_4 - N_3 = 57$.
Enfin, après $N_4 = 246$, le premier rang à partir duquel $u_n \in \left]0.2 + 14 * \frac{0.8-0.2}{26}, 0.2 + 15 * \frac{0.8-0.2}{26}\right[$ est $N_5 = 288$ donc on retient $N_5 - N_4 = 42$.
Le message codé est alors $[56, 88, 45, 57, 42]$

Exercice 1. Avec la même clé de chiffrement, $\lambda = 4, u_0 = 0.6, x1 = 0.2$ et $x2 = 0.8$, coder les messages suivants

1. 'BONJOUR'
2. 'AAAA'

Exercice 2. En faisant varier le terme u_0 de la clé de chiffrement, coder les mêmes messages que les messages précédents. Vérifier qu'une petite variation sur cette donnée se traduit par un message codé bien différent.

Déchiffrement.

Ensuite on considère le message codé qui est ici une liste de nombres $[N_1, N_2 - N_1, \ldots, N_p - N_{p-1}]$.

On itère la suite logistique avec la clé de chiffrement λ , u_0, $[x1, x2]$. On note les intervalles dans lesquels se trouvent u_{N_1}, u_{N_2}, ..., u_{N_p} et on retient les lettres correspondants à ces intervalles. On trouve alors le message déchiffrer.

Exercice 3. Avec la clé de chiffrement $\lambda = 4, u_0 = 0.6, x1 = 0.2$ et $x2 = 0.8$, décoder les messages suivants

1. $[56, 88, 45, 57, 42]$
2. $[70, 74, 102, 80, 69]$
3. Que constatez-vous ?

Exercice 4. Effectuer d'autres essais de déchiffrement, en particulier pour les messages précédemment codés .

Amélioration.

Notre méthode de chiffrement repose sur 2 a priori :

- ☞ l'idée que les clés de chiffrement doivent rester secrètes... ce qui n'est malheureusement pas garanti
- ☞ l'idée qu'un espion découvrant un message codé ne peut pas déterminer le message d'origine (et non plus les clés de chiffrement)

Ceci est bien garanti par la méthode. Il y a cependant un souci : si on veut utiliser de façon régulière ce type de codage, il est important de ne pas changer trop souvent de clés de chiffrement (par peur des interceptions ... ou pour pouvoir communiquer dans n'importe quelle circonstance, y compris les plus délicates où il n'est pas possible de prévenir le changement de clés.
Aussi comme notre message est codé toujours de la même façon, ou en tout cas, deux messages commençant de la même façon seront codés par des messages secrets ayant également le même début.
En particulier, si un espion récupère deux messages débutant de la même façon, il peut en déduire certaines choses sur les premiers mots utilisés ce qui peut donner des indications sur les premiers termes voire LE premier terme de la suite logistique utilisée.
Pour pallier à ce probleme, Batista a constaté que pour une clé de chiffrement donné et un message en clair donné, plusieurs listes correspondant à des codes secrets sont décodées par le message en clair.
Aussi Batista propose que pour coder une lettre d'un message, lorsque l'on arrive sur le 'bon' intervalle, on tire un nombre au hasard. Si ce nombre est supérieur à une valeur prédéfinie, on retient le rang du terme de la suite et sinon on continue le processus jusqu'à revenir sur cet intervalle ; on refait alors un tirage au sort...