Projet Python : Chaos déterministe

Sujet proposé par Julien Guillod : julien.guillod@upmc.fr

1 Introduction

Le but de ce projet est d'étudier un modèle extrêmement simple d'évolution d'une population. Malgré son caractère simpliste, ce modèle présente de nombreuses propriétés très intéressantes et quelques unes seront étudiées lors de ce projet.

La **proportion d'une population** à un temps discret $i \in \mathbb{N}$ est notée $x_i \in [0, 1]$. L'évolution de la population est donnée par $x_{i+1} = f_\mu(x_i)$ où $f_\mu : [0, 1] \to [0, 1]$ est la fonction définie par :

$$f_\mu(x) = \mu x(1 - x).$$

Le paramètre $\mu \in [0, 4]$ décrit la croissance de la population. L'application f_μ est appelée l'application logistique de paramètre μ.

L'intuition derrière le modèle est la suivante : lorsque la population x est petite, sa croissance est donnée par μx ; lorsque la population est proche du maximum $x = 1$ alors sa croissance est quasiment zéro car il n'y a plus de place pour tout le monde.

Pour produire les représentations graphiques, `matplotlib` sera utilisé. De plus, pour représenter des vecteurs et des matrices, `numpy` peut être utile même si son utilisation n'est pas indispensable pour la réalisation de ce projet. L'ensemble du projet est réalisable en moins de 75 lignes de code.

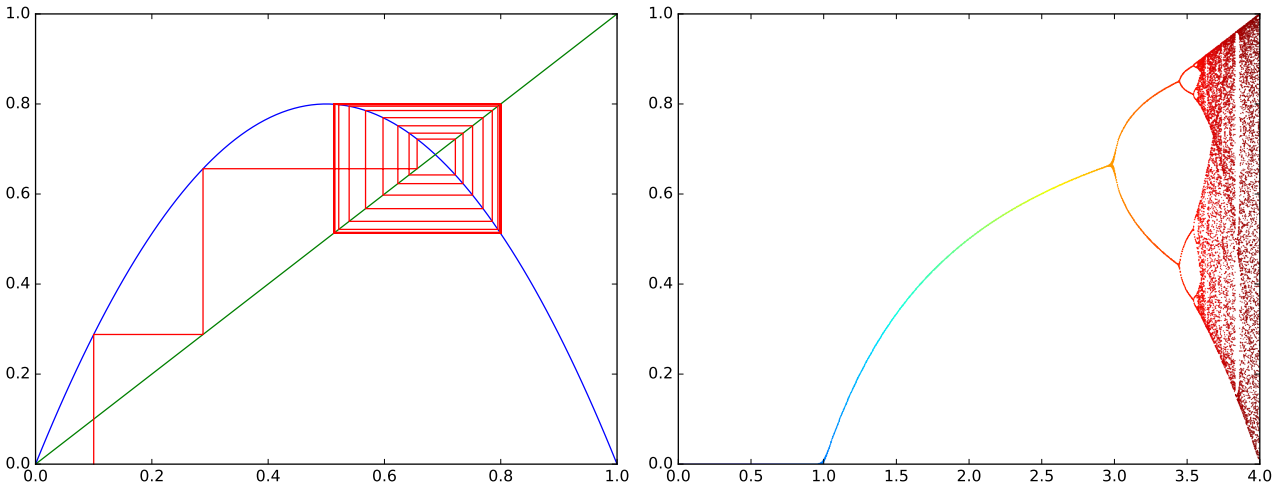

FIGURE 1 – (a) Diagramme araignée pour $x_0 = 0.1$ et $\mu = 3.2$. La fonction f_μ est représentée en bleu et la fonction identité en vert. Le diagramme araignée représente la suite des populations x_i commençant à partir de $x_0 = 0.1$. Cette figure sera construite au §4. (b) Diagramme de bifurcation représentant μ sur l'axe horizontal et sur l'axe vertical les valeurs prises par la population x_i en temps grand (*i.e.* lorsque i est grand). Cette figure sera construite au §5.

2 Définition de la fonction f_μ

Dans cette section, la fonction f_μ est définie et étudiée.

1. Définir la fonction f_μ en python comme `f(x, mu)` $= f_\mu(x)$.
2. En utilisant `matplotlib` (et `numpy` si besoin) définir une fonction `plot_f(mu)` qui trace le graphe de la fonction f_μ et tester cette fonction pour plusieurs valeurs de μ.
3. Pourquoi la valeur de μ ne peut pas être supérieur à 4 ?

3 Simulation

Le but de cette section est de calculer la population x_i.

1. Définir une fonction qui prend comme argument une valeur de μ, une donnée initiale $x_0 \in [0, 1]$, et un nombre $n \in \mathbb{N}$ et retourne la liste $S_\mu^n = (x_0, x_1, x_2, \dots, x_n)$.
2. Modifier fonction précédente pour quelle puisse prendre un paramètre optionnel $m \in \mathbb{N}$ et retourne la liste $S_\mu^{m,n} = (x_m, x_{m+1}, \dots, x_n)$, c'est-à-dire retire les $m-1$ premiers éléments à S_μ^n.
3. Tester cette fonction en faisant la représentation graphique de la liste S_μ^n pour différentes valeurs des paramètres x_0 et μ.
4. Étudier ce qui se passe qualitativement lorsque μ varie ? Et lorsque x_0 varie ? Déterminer quels sont les différents comportements observés ? Quel est le comportement lorsque μ est très proche de la valeur maximale 4 ?

4 Diagramme araignée

Une façon d'étudier plus spécifiquement ce qui se passe lorsque μ varie est de faire un diagramme dit araignée (voir figure 1a). La construction de ce diagramme se fait de la façon suivante :

a. A partir du point $(x_0, 0)$ tracer une ligne verticale jusqu'au point $(x_0, f_\mu(x_0)) = (x_0, x_1)$;
b. Depuis ce point, tracer la ligne horizontale jusqu'à la diagonale, *i.e.* jusqu'au point $(f_\mu(x_0), f_\mu(x_0)) = (x_1, x_1)$;
c. Ensuite tracer la ligne verticale jusqu'à l'intersection avec le graphe de f_μ *i.e.* jusqu'au point $(x_1, f_\mu(x_1)) = (x_1, x_2)$;
d. Répéter les étapes b. et c. récursivement.

Le diagramme araignée consiste donc à relier par des droites les points

$$\big\{(x_0, 0), (x_0, x_1), (x_1, x_1), (x_1, x_2), (x_2, x_2), \dots, (x_n, x_n), (x_n, x_{n+1})\big\}.$$

1. Définir une fonction qui retourne la liste des points nécessaire pour construire le diagramme araignée. A noter qu'en python, il est parfois plus simple (notamment pour utiliser ensuite `matplotlib`) de représenter une liste de points $\big\{(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)\big\}$ comme le vecteur des abscisses et le vecteur des ordonnées, *i.e.* $(x_0, x_1, \dots, x_n)$ et $(y_0, y_1, \dots, y_n)$.
2. Définir une fonction qui trace le graphe de la fonction f_μ, le graphe de la fonction identité, ainsi que les segments reliant les points de la liste précédente.
3. En expérimentant, étudier qualitativement les effets des paramètres μ et x_0. Décrire le comportement observé lorsque :
   (a) $0 < \mu < 3$;
   (b) $3 < \mu < 3.4494897$;
   (c) $3.4494897 < \mu < 3.5440903$;
   (d) $\mu > 3.5440903$.

5 Diagramme de bifurcation

Les expérimentations précédentes laissent penser que le comportement de la suite x_i en temps grand (*i.e.* lorsque i est grand) est indépendant du choix de la condition initiale x_0 mais dépend beaucoup de la valeur du paramètre μ. Le but de cette section est de représenter graphiquement pour chaque valeur de μ l'ensemble des points $S_\mu^{m,n} = (x_m, x_{m+1}, \dots, x_n)$, *i.e.* en mettant μ sur l'axe horizontal et toutes les valeurs de $S_\mu^{m,n}$ sur l'axe vertical (voir figure 1b). Un bon choix de paramètres pour nettoyer le diagramme et garder seulement le comportement en temps long du système est $n = 200$ et $m = 100$ par exemple.

1. Définir une fonction qui donné une liste de valeurs de μ, $L = (\mu_0, \mu_1, \cdots, \mu_N)$, une donnée initiale $x_0 \in [0, 1]$ et des entiers $m, n \in \mathbb{N}$ retourne la liste des points $\left\{(\mu, x) : x \in S_\mu^{m,n} \text{ pour } \mu \in L\right\}$. A remarquer qu'il peut être judicieux comme au point 4.1 de retourner la liste des abscisses et celle des ordonnées plutôt que la liste des points.
2. Définir une fonction qui représente graphiquement cette liste de points. Il est suggéré de prendre pour L une liste de 1000 valeurs dans l'intervalle $[0, 4]$ et d'utiliser la fonction `scatter` de `matplotlib` avec les paramètres `edgecolor='None'` et `s=1`.
3. Interpréter le diagramme obtenu, en particulier ce qu'il dit sur le comportement en temps long du système. Déterminer approximativement pour quelles valeurs de μ le système :
   (a) possède zéro comme unique point fixe ;
   (b) possède un unique point fixe non-nul ;
   (c) oscille entre deux valeurs distinctes (cycle de longueur deux) ;
   (d) oscille entre quatre valeurs distinctes (cycle de longueur quatre) ;
   (e) oscille entre trois valeurs distinctes (cycle de longueur trois).

6 Représentation de l'attracteur (facultatif)

Pour des valeurs de μ proches de 4, les valeurs de la population x_i ont l'air d'être plus ou moins aléatoires. Pourtant, le système est purement déterministe dans le sens où donné une valeur initiale x_0, la population x_i est définie sans aléa. Ce comportement *a priori* aléatoire est appelé **chaos déterministe**. Pour bien remarquer que les points x_i sont pas déterminés de manière aléatoire, le but est de représentation graphiquement les points (x_n, x_{n+1}) et de voir que x_{n+1} n'est pas du tout aléatoire par rapport à x_n.

1. Pour chaque valeur de μ fixée, définir une fonction qui retourne la liste de points
   $$\left\{(x_m, x_{m+1}), (x_{m+1}, x_{m+2}), \dots, (x_n, x_{n+1})\right\}$$
   et représenter cette liste graphiquement pour différentes valeurs de μ.
2. Représenter graphiquement des points pour différentes valeurs de μ. Par exemple $n = 5000$ et $m = 100$ sont un bon choix de paramètres.
3. Comment serait le graphique précédent si chaque x_i était tiré aléatoirement dans l'intervalle $[0, 1]$ indépendamment de x_{i-1} ?