Divergence des trajectoires proches

Dans tout ce développement, on travaillera avec la transformation logistique de paramètre 4 :

$$f_4 : \left\{ \begin{array}{rcl} [0,1] & \to & [0,1], \\ x & \mapsto & 4x(1-x). \end{array} \right.$$

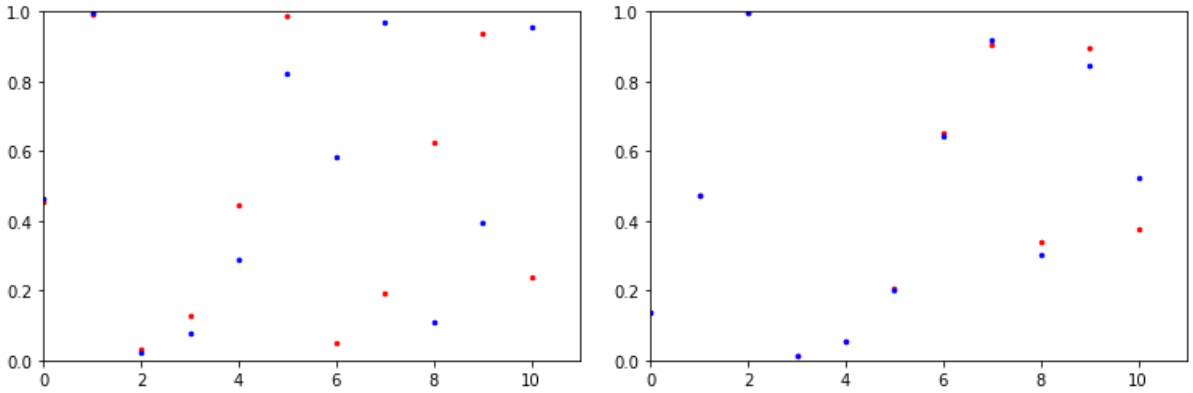
Étant donné un point $x \in [0,1]$, on définit par récurrence une suite $(x_n)_{n\geq 0}$:

$$\left\{ \begin{array}{rcl} x_0 & = & x\,; \\ x_{n+1} & = & f_4(x_n) \quad \forall n \geq 0. \end{array} \right. \tag{0.1}$$

La transformation f_4 est chaotique : pour la plupart des valeurs de x, le comportement de la suite $(x_n)_{n\geq 0}$ est très compliqué. Nous allons introduire une des raisons pour cela (et une des définitions du chaos) : la *divergence des trajectoires proches*.

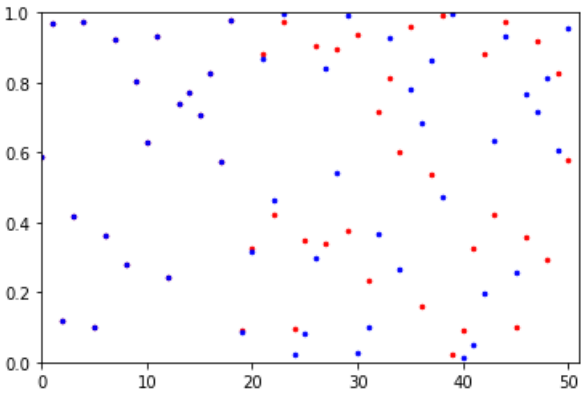
1 Point de vue qualitatif

Nous allons utiliser l'outil traçant le graphe de la suite. Le diagramme en toile d'araignée n'est pas très utile pour les transformations chaotiques, et l'histogramme est discuté dans le document "Ergodicité de transformations chaotiques". Nous allons cependant modifier la fonction `suite_graphe` pour afficher le graphe de deux suites $(x_n)_{n\geq 0}$ et $(y_n)_{n\geq 0}$ construites par récurrence à partir de deux valeurs initiales x et y proches.



À gauche : graphe des 10 premières valeurs de $(x_n)_{n\geq 0}$ (en rouge) et de $(y_n)_{n\geq 0}$ (en bleu), avec $|y-x| = 10^{-2}$. À droite : de même, avec $|y-x| = 10^{-3}$.

Comme on l'observe, les deux trajectoires restent proches pendant un certain temps, mais finissent par se séparer. Plus les valeurs de départ sont proches, plus les deux suites mettront de temps à se séparer, mais elles finiront bien par se séparer. De plus, à partir du moment où les deux suites sont assez éloignées l'une de l'autre, leurs comportements vont devenir essentiellement indépendants : la connaissance des valeurs de l'une apportera très peu d'information sur l'autre. Cette dichotomie entre une plage de temps sur laquelle les suites sont proches et une plage de temps sur laquelle les suites sont indépendantes est mieux visible en prenant des valeurs initiales plus proches et un plus grand nombre d'itérations.



Graphe des 50 premières valeurs de $(x_n)_{n\geq 0}$ (en rouge) et de $(y_n)_{n\geq 0}$ (en bleu), avec $|y-x| = 10^{-8}$.

Ce phénomène a une conséquence du point de vue des simulations. Le calcul informatique introduit un certain nombre d'erreurs : erreur d'arrondis de la valeur initiale x, qui est stockée en binaires ; erreur d'arrondi à chaque fois que l'on applique f_4... Mais faire une erreur d'arrondi sur x, c'est travailler à partir d'une valeur y, proche de x mais différente. Par conséquent, à partir d'un certain rang, les valeurs de x_n calculées par notre programme seront très différentes des valeurs exactes !

Cela a des conséquences pratiques importantes. La plus célèbre s'applique aux prévisions météorologiques. L'évolution de l'état de l'atmosphère est chaotique. Les données que l'on peut mesurer à un instant donné (température, pression, humidité, vent...) sont imprécises, nos modèles de l'atmosphère sont imprécis, et les calculs informatiques apportent aussi leur lot d'approximations. Toutes ces imprécisions, ainsi que le caractère chaotique du système atmosphérique, font que nos modélisations vont s'écarter irrémédiablement de la réalité : après un temps relativement court, de quelques jours, les prévisions météorologiques ne sont plus fiables.

2 Point de vue quantitatif

Cette divergence des trajectoires proches peut être étudiée quantitativement. Le principe sous-jacent, énoncé de façon peu rigoureuse, est le suivant. Supposons que f_4 admette une "dérivée moyenne" $\lambda > 1$. Si x et y sont proches, alors :

$$f_4(x) - f_4(y) \simeq (f_4(y) + \lambda(x-y)) - f_4(y) = \lambda(x-y).$$

Si $f_4(x)$ et $f_4(y)$ sont proches, on peut itérer cette estimation. On obtient récursivement, tant que x_n et y_n sont proches,

$$x_n - y_n \simeq \lambda^n (x-y).$$

Les trajectoires divergent donc à vitesse exponentielle. Multiplier par 2 la précision initiale n'apporte que quelques étapes supplémentaires pendant lesquelles les deux suites sont proches. On peut essayer d'estimer, dans notre cas, la constante λ. Pour cela, on va choisir x et y très proches, et on va mesurer le temps à partir duquel les trajectoires correspondantes sont éloignées. Pour cela, on va utiliser le module Decimal, qui permet de travailler en précision arbitraire, et utiliser un algorithme de seuil : dès que la distance entre les deux suites dépasse le seuil arbitraire de $0,1$, nous les considérons suffisamment séparées, et nous arrêtons l'algorithme.

Un exemple de tel algorithme est fourni dans la dernière partie. D'après cet algorithme, si l'on choisit x et y tels que $|x-y| = 10^{-100}$, il faut environ 329 itérations pour leurs trajectoires se séparent. On remarque d'ailleurs que ce nombre d'itérations dépend peu de la valeur initiale de x et y, pourvu que leur écart reste le même. Cela nous donne

$$0,1 \simeq |x_{329} - y_{329}| \simeq \lambda^{329}|x-y| = \lambda^{329} 10^{-100},$$

soit $\lambda \simeq 10^{\frac{99}{329}} \simeq 1,999$. Ce nombre est très proche de la valeur théorique de λ, que l'on peut calculer explicitement pour f_4, et qui est de 2.

La valeur $\ln(\lambda)$, valant ici $\ln(2)$, est appelée *exposant de Lyapunov* de f_4. On dit parfois que les transformations chaotiques sont celles ayant un exposant de Lyapunov strictement positif ; cela revient à dire que $\lambda > 1$, et donc que les trajectories proches se séparent à vitesse exponentielle.

3 Code de l'algorithme de seuil

```
import numpy
from decimal import *
getcontext().prec = 100

# Definition de la fonction

def fonction (x) :
   return 4*x*(1-x)


# Definition de l'algorithme de seuil
# Valeur_1, Valeur_2 : valeurs initiales des suites. Variables de type Decimal.
# Seuil : seuil au-dessus duquel l'algorithme s'arrete et ressort le nombre d'iterations. Variable
    de type Decimal.

def algorithme_seuil (Valeur_1, Valeur_2, Seuil) :
   x = Valeur_1
   y = Valeur_2
   Iterations = 1000
   for n in range(Iterations) :
      x = fonction(x)
      y = fonction(y)
      if abs(y-x) > Seuil :
         return "Les trajectoires se sont separees apres " + str(n+1) + " iterations."
   return "Apres " + str(Iterations) + " iterations, les deux trajectoires ne se sont pas separees."


# Application de l'algorithme
# Ecart entre les valeurs initiales de 10^(-p), seuil de 0,1.

Seuil = Decimal(0.1)
p = 10

x = Decimal(numpy.random.uniform(0,1))
y = x+Decimal(10)**(-p)

print( algorithme_seuil(x, y, Seuil) )
```