Étude de la suite logistique

1. Tracé de graphes avec Python

En liaison avec le module `numpy`, le module `matplotlib.pyplot` contient un certain nombre de fonctions dédiées au tracé de graphes. La première chose à faire consiste donc à importer ces deux modules :

```
import numpy as np
import matplotlib.pyplot as plt
```

Les fonctions du module `numpy` sont désormais utilisables à condition d'être préfixées par `np` ; celles du module `matplotlib.pyplot` par le préfixe `plt`.
Le principe général d'un tracé de graphe consiste, à partir d'une figure initialement vide, à lui ajouter progressivement des éléments graphiques (graphes, axes, légendes, ...) qui vont se superposer. On crée une figure vide à l'aide de la fonction `plt.figure()`. Celle-ci possède un certain nombre de paramètres qui déterminent son apparence (la plupart du temps les valeurs par défaut sont suffisantes) ; on retiendra surtout le paramètre `figsize` qui permet de fixer la taille (en inches) de la fenêtre graphique.
Viennent ensuite les différentes fonctions qui ajoutent des éléments graphiques et que nous allons détailler ci-dessous puis enfin la fonction `plt.show()` qui permet de visualiser la figure dans une fenêtre annexe.

Tracé simple

Si $X = [x_0, x_1, \ldots, x_n]$ et $Y = [y_0, y_1, \ldots, y_n]$ sont deux listes numériques de même taille, la fonction `plt.plot(X, Y)` ajoute à la figure une ligne brisée qui relie les points de coordonnées (x_i, y_i) pour $i \in [\![0, n]\!]$.
Tracez votre première figure en recopiant le code ci-dessous :

```
plt.figure()

X = [np.cos(t) for t in range(20)]
Y = [np.sin(t) for t in range(20)]

plt.plot(X, Y)

plt.show()
```

Vous pouvez observer que des axes ont été automatiquement ajoutés et ajustés de façon à ce que le graphe remplisse la figure dans son entier.

Le choix d'une figure rectangulaire n'est sans doute pas adapté au tracé que nous venons de réaliser. Recommencez en changeant la première ligne par :

```
plt.figure(figsize=(8, 8))
```

pour obtenir le tracé dans une fenêtre carrée.

Remplacez enfin le nombre de points tracés (20) par une valeur plus grande, par exemple 256, et observez le résultat obtenu.

Tracé du graphe d'une fonction

Pour tracer le graphe d'une fonction d'équation $y = f(x)$ sur l'intervalle $[a, b]$ il suffit de subdiviser cet intervalle avec un pas suffisamment fin $x_0 = a < x_1 < \cdots < x_{n-1} = b$ puis pour chaque valeur x_i de cette subdivision calculer la valeur $y_i = f(x_i)$. La fonction `linspace` du module `numpy` réalise simplement cette subdivision :

`np.linspace(a, b, n)` subdivise $[a, b]$ en n points équirépartis (le pas de la subdivision est donc égal à $\frac{b-a}{n-1}$).

Recopier le script ci-dessous :

```
plt.figure()

X = np.linspace(-np.pi, np.pi, 256)
C = [np.cos(x) for x in X]
S = [np.sin(x) for x in X]

plt.plot(X, C)
plt.plot(X, S)

plt.show()
```

Observez que les deux graphes ont été automatiquement tracés en deux couleurs différentes. Il est bien sur possible de choisir la couleur du graphe, ainsi que l'épaisseur du trait et le style du tracé. Recommencez en remplaçant les deux tracés par :

```
plt.plot(X, C, color="blue", linestyle="-", linewidth=2.5)
plt.plot(X, S, color="red", linestyle="dashed")
```

Donner ici l'ensemble des possibilités que vous offrent les paramètres optionnels serait fastidieux ; reportez-vous à l'aide associée à la fonction `plot` pour les connaître : `help(plt.plot)`. Notons simplement que des raccourcis existent pour les options les plus communes ; par exemple, les deux tracés ci-dessus peuvent être obtenus en écrivant plus succinctement :

```
plt.plot(X, C, 'b-', linewidth=2.5)
plt.plot(X, S, 'r--')
```

Ajouter une légende

On peut aussi ajouter une légende associée à chacune des courbes tracées en utilisant le paramètre `label` de la fonction `plot`. La fonction `legend` associe à la couleur du tracé le label que vous lui avez donné :

```
plt.plot(X, C, color="blue", linestyle="-", label='cos')
plt.plot(X, S, color="red", linestyle="dashed", label='sin')
plt.legend(loc='upper left')
```

Notons qu'il est possible de placer une marque aux endroits des points (x_i, y_i). Observer le résultat du code suivant (on diminue le pas de la subdivision pour plus de lisibilité) :

```
plt.figure()

X = np.linspace(-np.pi, np.pi, 32)
C = [np.cos(x) for x in X]
S = [np.sin(x) for x in X]

plt.plot(X, C, marker='o', linestyle='')    # on pourrait écrire plt.plot(X, C, 'o')
plt.plot(X, S, marker='s', color='red')     # on pourrait écrire plt.plot(X, C, 'rs-')

plt.show()
```

Modifier les axes

Les fonctions `plt.xlim` et `plt.ylim` permettent de choisir les valeurs extrêmes des axes. Par exemple, pour éviter que les graphes que nous venons de tracer touchent les bords du cadre, on peut augmenter légèrement l'axe des ordonnées :

```
plt.ylim(-1.1, 1.1)
```

Les fonctions `plt.xticks` et `plt.yticks` contrôlent les subdivisions des axes en prenant pour paramètre la subdivision souhaitée. Par exemple :

```
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.yticks([-1, 0, 1])
```

La fonction `plt.grid` permet de faire apparaitre le quadrillage associé à la subdivision choisie. Enfin, la fonction `plt.title` ajoute un titre à la figure créée :

```
    plt.grid()
    plt.title('Les fonctions cos et sin')
```

2. Étude de la suite logistique

Noua allons maintenant utiliser les fonctions graphiques de Python pour observer quelques particularités de la suite $(x_n)_{n\in\mathbb{N}}$ définie par la donnée de la valeur $x_0 \in]0,1[$ et de la relation de récurrence : $x_{n+1} = \mu x_n(1-x_n)$, où μ est une constante de l'intervalle $[0,4]$.
On peut visualiser le comportement de ces suites à l'aides de graphes tels ceux tracés en figure 1, en positionnant l'entier n sur l'axe des abscisses et x_n sur l'axe des ordonnées.

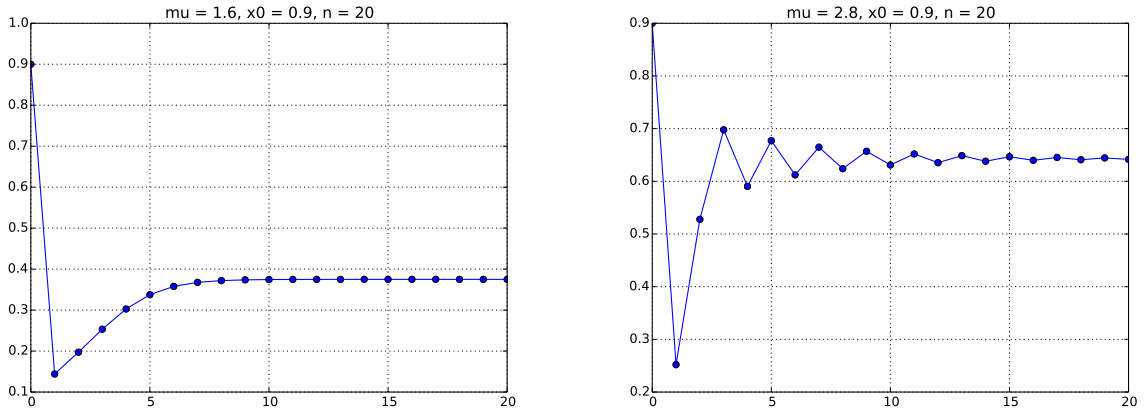

Figure 1: Pour $\mu = 1{,}6$ la suite (x_n) est croissante à partir d'un certain rang, pour $\mu = 2{,}8$ les suites (x_{2n}) et (x_{2n+1}) sont (à partir d'un certain rang) adjacentes. Dans les deux cas la suite x converge.

Une autre façon de visualiser le comportement asymptotique de la suite $(x_n)_{n\in\mathbb{N}}$ consiste, à partir des graphes de la fonction $f_\mu : x \mapsto \mu x(1-x)$ et de la droite d'équation $y = x$, à tracer la ligne brisée reliant les points de coordonnées (x_0,x_1), (x_1,x_1), (x_1,x_2) , (x_2,x_2), (x_2,x_3), (x_3,x_3), ... , (x_{n-1},x_{n-1}), (x_{n-1},x_n) (illustration figure 2).

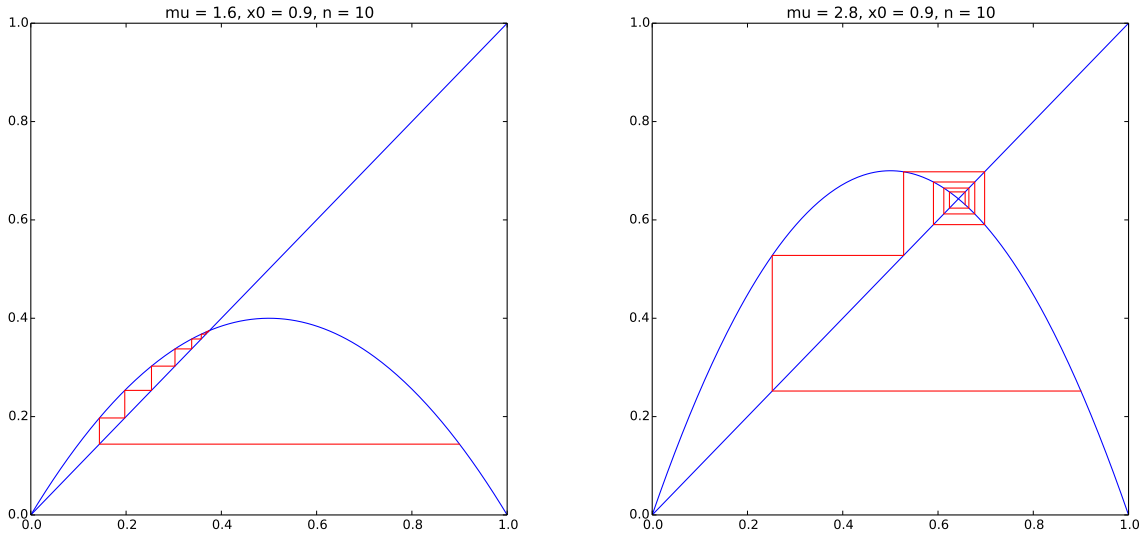

Figure 2: Une autre représentation des suites obtenues pour $\mu = 1{,}6$ et $\mu = 2{,}8$.

Question 1. Définir une fonction `logistique1(mu, x0, n)` qui permette le tracé des graphes que l'on trouve figure 1, puis une fonction `logistique2(mu, x0, n)` pour le tracé des graphes de la figure 2.

Question 2. À l'aide de ces deux fonctions nous allons (sans démonstration) postuler le comportement de la suite $(x_n)_{n\in\mathbb{N}}$ en fonction de la valeur de μ. On choisira à chaque fois pour valeur de départ $x_0 = 0{,}9$ (bien qu'en réalité et mis à part quelques valeurs particulières cette valeur importe peu) et on essayera plusieurs valeurs de n.

a. Lorsque $\mu \in [0,1]$, observer que la suite $(x_n)_{n\in\mathbb{N}}$ converge vers 0.

b. Lorsque $\mu \in [1,3]$, observer que la suite $(x_n)_{n\in\mathbb{N}}$ converge vers le point fixe $\dfrac{\mu-1}{\mu}$. Quelle différence peut-on faire suivant que $\mu \in [1,2]$ ou que $\mu = [2,3]$?

c. Lorsque $\mu = 3{,}05$, qu'observe-t-on ? Et lorsque $\mu = 3{,}5$?

d. Observer enfin la situation pour $\mu = 3{,}86$.

Il est possible de démontrer que lorsque μ est compris entre 3 et $1+\sqrt{6}$ (environ $3{,}45$) la suite $(x_n)_{n\in\mathbb{N}}$ finit par osciller entre deux valeurs dépendantes de μ mais pas de x_0. Entre $3{,}45$ et (environ) $3{,}54$ la suite finit par osciller entre quatre valeurs. au delà de cette valeur la suite oscille entre huit valeurs, puis seize, etc. À partir de d'environ $3{,}57$, le chaos s'installe et mis à part certaines valeurs il n'est plus possible de décrire le comportement asymptotique de la suite.
Tout ceci peut être résumé par le *diagramme des bifurcations* : l'axe des abscisses porte les valeurs du paramètre μ, l'axe des ordonnées les valeurs d'adhérences possibles.

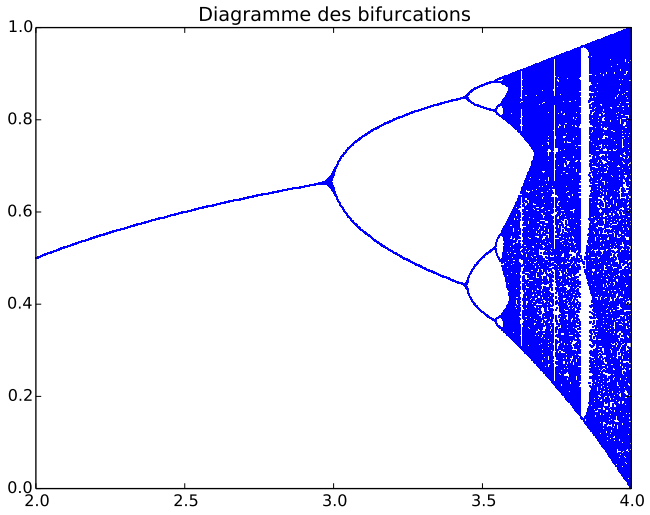

Figure 3: Le diagramme des bifurcations de la suite logistique.

Question 3. Le diagramme des bifurcations de la figure 3 a été obtenu en faisant varier μ entre 2 et 4 avec un pas égal à $0{,}002$. Pour chacune des valeurs de μ sont représentées en ordonnée les valeurs distinctes à 10^{-3} près de $x_{101}, x_{102}, \ldots, x_{200}$.
Le tracé a été obtenu avec les options `marker=',', linestyle=''`.
Rédiger un script Python qui réalise le tracé de ce diagramme.

Exposant de Lyapunov

Le système dynamique obtenu pour $\mu = 4$ est chaotique : une infime variation de la valeur initiale x_0 modifie du tout au tout la valeur de x_n après seulement quelques itérations. Pour mesurer l'influence d'un écart e_0 sur x_0 pour la suite $(x_n)_{n\in\mathbb{N}}$ on calcule l'écart absolu après une itération : $e_1 = f_\mu(x_0+e_0) - f_\mu(x_0)$. L'écart relatif vaut $\dfrac{|e_1|}{|e_0|} = \left|\dfrac{f_\mu(x_0+e_0)-f_\mu(x_0)}{e_0}\right|$, quantité que l'on approche par $|f'_\mu(x_0)|$ sachant que e_0 est petit.
Après n itérations l'écart relatif vaut donc :

$$\frac{|e_n|}{|e_0|} = \frac{|e_n|}{|e_{n-1}|} \times \cdots \times \frac{|e_1|}{|e_0|} \approx \prod_{k=0}^{n-1} |f'_\mu(x_k)|.$$

Notre objectif est de déterminer si les écarts s'amplifient et donc si ce produit est supérieur à 1 ou si le système est stable et le produit inférieur à 1. Sachant que réaliser une addition est plus rapide qu'une multiplication, on étudie plutôt le logarithme de cette quantité :

$$\prod_{k=0}^{n-1} |f'(x_k)| < 1 \iff \sum_{k=0}^{n-1} \ln |f'_\mu(x_k)| < 0$$

et pour relativiser le rôle du choix de n on divise cette somme par n.
Ceci conduit à définir l'exposant de Lyapunov :

$$\lambda_\mu = \lim_{n\to+\infty} \frac{1}{n} \sum_{k=0}^{n-1} \ln |f'_\mu(x_k)|.$$

Question 4. Rédiger une fonction `lyapunov(mu, x0, n)` qui calcule une approximation de cette quantité en fonction de la valeur de μ, de x_0 et du nombre d'itérations n choisi.
Tracer ensuite le graphe représentant les variations de λ_μ en fonction de μ dans l'intervalle $[3,4]$ à partir de la valeur $x_0 = 0{,}9$.

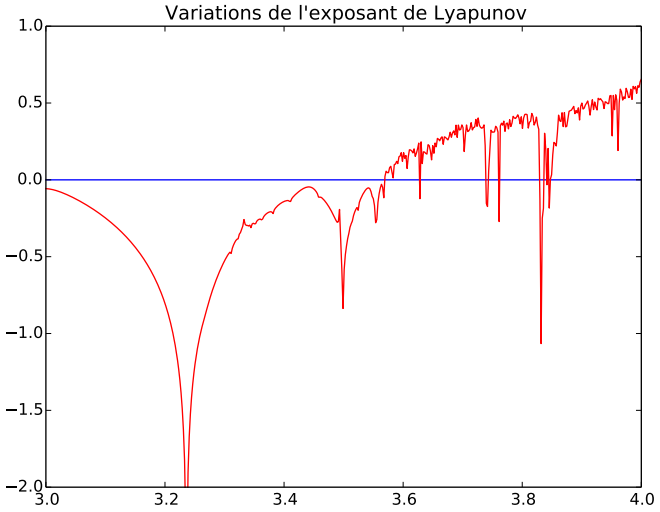

Figure 4: Les variations de l'exposant de Lyapunov entre 3 et 4.

Comment interpréter ce graphique ?

Et pour les plus rapides

Observons de nouveau le diagramme des bifurcations. Pour $\mu_0 = 3$ a lieu la première bifurcation ; pour $\mu_1 \approx 3{,}45$ a lieu la seconde. Plus généralement on note μ_k la valeur de μ où a lieu la $(k+1)^e$ bifurcation. La constante de Feigenbaum est la valeur de la limite :

$$\delta = \lim_{n\to+\infty} \frac{\mu_{n+1} - \mu_n}{\mu_{n+2} - \mu_{n+1}}.$$

Question 5. Essayer d'obtenir une approximation de cette constante (pour information on a $\delta = 4{,}669\,201\cdots$).

Indication. Pour déterminer la taille du cycle limite, définir une fonction qui calcule le nombre de valeurs distinctes à 10^{-4} près des termes $x_{n+1},\ldots,x_{n+256}$ pour une grande valeur de n (par exemple 100 000). En procédant à des recherches dichotomiques déterminer les six premières bifurcations et calculer $\dfrac{\mu_5 - \mu_4}{\mu_6 - \mu_5}$.

Remarque. Cette constante a ceci de remarquable qu'on la retrouve dans d'autres diagrammes des bifurcations, par exemple en étudiant les suites définies par la relation de récurrence $x_{n+1} = \mu \sin(x_n)$ ou encore $x_{n+1} = \mu - x_n^2$. Si vous en avez le temps vous pouvez mettre en évidence son caractère universel en reprenant votre étude avec l'une ou l'autre de ces deux relations.