Groupe A : Nuages de points

Programmes à construire

On fixe un paramètre $c \in [0,4]$. Soit $f_c : [0,1] \to [0,1]$ la fonction définie par

$$f_c(x) = cx(1-x) \text{ pour tout } x \in [0,1].$$

L'intervalle $[0,1]$ est stable par f_c. Étant donnée une valeur $x_0 \in [0,1]$, on définit par récurrence la suite $(u_n)_{n\geq 0}$ telle que $u_0 = x_0$ et $u_{n+1} = f_c(u_n)$ pour tout $n \geq 0$.

Objectif 1 : Écrivez une fonction, appelée liste_de_points , qui prend en entrée un entier n et un réel $x_0 \in [0,1]$, et retourne la liste des couples (k, u_k) pour $0 \leq k < n$.

Objectif 2 : Écrivez une fonction, appelée nuage_de_points, qui prend en entrée un entier n et un réel $x_0 \in [0,1]$, et affiche les n premiers points (k, u_k) sous la forme d'une nuage de points.

Objectif 3 : Reprenez la fonction nuage_de_points, et écrivez une fonction nuage_de_points_double qui affiche simultanément les nuages de points correspondants à deux conditions initiales x_0 et y_0 et un même entier n, en deux couleurs différentes de votre choix.

Étude de suites logistiques

À l'aide des programmes précédents, remplissez le tableau joint.

Pour aller plus loin, vous pourrez notamment utiliser la fonction nuage_de_points avec des points x_0 et y_0 très proches, observer si leurs orbites restent proches, et le cas échéant, estimer le temps de séparation.

Quelques commandes utiles

Ici, un point est un couple de coordonnées (x,y).

Soit liste1 une liste de points.

- ▷ L'objet graphique = points(liste1) est le graphique contenant le nuage de points constitué des points de la liste.
- ▷ La commande show(graphique) affiche l'objet graphique.

On peut rajouter des options dans la fonction points, qui permettent notamment de fixer la fenêtre de visualisation ou la couleur des points. Utilisez la commande points? pour afficher ces différentes options.

On peut additionner des graphiques. graphique1 + graphique2 contient à la fois les objets de graphique1 et ceux de graphique2.

Groupe B : Valeurs précises de la suite

Programmes à construire

On fixe un paramètre $c \in [0,4]$. Soit $f_c : [0,1] \to [0,1]$ la fonction définie par

$$f_c(x) = cx(1-x) \text{ pour tout } x \in [0,1].$$

L'intervalle $[0,1]$ est stable par f_c. Étant donnée une valeur $x_0 \in [0,1]$, on définit par récurrence la suite $(u_n)_{n\geq 0}$ telle que $u_0 = x_0$ et $u_{n+1} = f_c(u_n)$ pour tout $n \geq 0$.

Question 1 : Si on approche un nombre réel avec N *bits* de précision, combien de décimales exactes a-t-on, typiquement ? Quelle valeur de N choisir pour avoir 100 décimales de précision ?

Objectif 2 : Écrivez une fonction, appelée suite_de_valeurs, qui prend en entrée un entier n et un réel $x_0 \in [0,1]$, et affiche les unes à la suite des autres les n premières valeurs de la suite $(u_k)_{k\geq 0}$.

Objectif 3 : Reprenez le programme précédent, et améliorez sa précision à au moins 100 décimales.

Objectif 4 : Reprenez le programme précédent, mais au lieu d'afficher les valeurs de la suite, enregistrez-les dans un fichier texte nommé ValeursSuite.txt.

Étude de suites logistiques

À l'aide des programmes précédents, remplissez le tableau joint.

Quand la suite $(u_n)_{n\geq 0}$ converge, déterminez numériquement et précisément sa vitesse de convergence.

Quelques commandes utiles

Pour que les réels aient une précision de N *bits*, ajoutez au début de votre programme la ligne RealNumber = RealField(N).

Pour afficher un nombre, on utilise la commande print. Utilisez la commande print? pour obtenir plus d'informations.

Pour écrire Hello world! dans le fichier Test.txt :

- ▷ Ouvrez le fichier. Pour cela, on crée une variable mon_fichier = open('Test.txt','w').
- ▷ Écrivez dans le fichier à l'aide de la commande mon_fichier.write('Hello world!'). Pour enregistrer un nombre, convertissez-le d'abord en une chaîne de caractère.
- ▷ Une fois que vous avez terminé d'écrire dans un fichier, fermez-le à l'aide de la commande mon_fichier.close().

Groupe C : Histogramme

Programmes à construire

On fixe un paramètre $c \in [0,4]$. Soit $f_c : [0,1] \to [0,1]$ la fonction définie par

$$f_c(x) = cx(1-x) \text{ pour tout } x \in [0,1].$$

L'intervalle $[0,1]$ est stable par f_c. Étant donnée une valeur $x_0 \in [0,1]$, on définit par récurrence la suite $(u_n)_{n\geq 0}$ telle que $u_0 = x_0$ et $u_{n+1} = f_c(u_n)$ pour tout $n \geq 0$.

Objectif 1 : Écrivez une fonction, appelée liste_de_valeurs , qui prend en entrée un entier n et un réel $x_0 \in [0,1]$, et retourne la liste des n premières valeurs de la suite $(u_k)_{k\geq 0}$.

Objectif 2 : Écrivez une fonction, appelée histogramme_de_valeurs, qui prend en entrée un entier n et un réel $x_0 \in [0,1]$, et affiche l'histogramme des n premières valeurs de la suite $(u_k)_{k\geq 0}$.

Question 3 : À quoi correspondent les hauteurs des rectangles de l'histogramme ? Rajoutez l'option density=true dans la fonction histogram. À quoi correspondent les hauteurs des rectangles dans le nouvel histogramme ?

Étude de suites logistiques

À l'aide des programmes précédents, remplissez le tableau joint.

Quelques commandes utiles

Soit liste1 une liste de réeels.

- ▷ L'objet graphique = histogram(liste1) est le graphique contenant l'histogramme des réels de la liste.
- ▷ La commande show(graphique) affiche l'objet graphique.

Par défaut, la classe la plus basse commence au nombre le plus petit de liste1 , et la classe la plus haute atteint le nombre le plus grand de liste1 .

On peut rajouter des options dans la fonction histogram. Utilisez la commande histogram? pour afficher ces différentes options.

Groupe D : Toile d'araignée

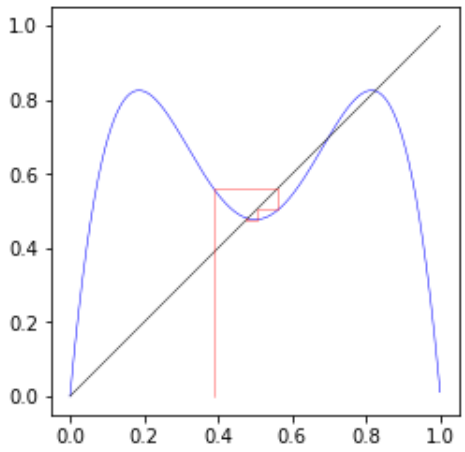
Programmes à construire

On fixe un paramètre $c \in [0,4]$. Soit $f_c : [0,1] \to [0,1]$ la fonction définie par

$$f_c(x) = cx(1-x) \text{ pour tout } x \in [0,1].$$

L'intervalle $[0,1]$ est stable par f_c. Étant donnée une valeur $x_0 \in [0,1]$, on définit par récurrence la suite $(u_n)_{n\geq 0}$ telle que $u_0 = x_0$ et $u_{n+1} = f_c(u_n)$ pour tout $n \geq 0$.

Le but, dans un premier temps, est de construire des **graphiques en toile d'araignée**, qui permettent de visualiser comme ci-contre une suite définie par récurrence.



Objectif 1 : La ligne rouge ci-dessus est une ligne brisée. Écrivez une fonction, appelée liste_de_points , qui prend en entrée un entier n et un réel $x_0 \in [0,1]$, et retourne la liste des points définissant cette ligne brisée jusqu'à la valeur u_n.

Objectif 2 : Écrivez une fonction, appelée toile_araignee , qui prend en entrée un entier n et un réel $x_0 \in [0,1]$, et affiche comme la ligne brisée rouge ci-dessus les n premières valeurs de la suite $(u_k)_{k\geq 0}$.

Objectif 3 : Complétez la fonction toile_araignee pour qu'elle affiche en plus le graphe de f_c (en bleu) ainsi que la droite d'équation $y = x$ (en noir). Affichez les objets dans le bon ordre pour une meilleure visibilité.

Étude de suites logistiques

À l'aide des programmes précédents, remplissez le tableau joint. Suivant les valeurs de c, vous pourrez remplacer f par $f^{(2)}$ ou $f^{(4)}$.

Quelques commandes utiles

Ici, un point est un couple de coordonnées (x,y).

Soit liste1 une liste de points.

- ▷ L'objet graphique = line(liste1) est la ligne brisée reliant les points successifs de liste1 .
- ▷ La commande show(graphique) affiche l'objet graphique.

Soit fonction1 une fonction prenant un réel et retournant un réel. L'objet graphique = plot(fonction1) est le graphe de fonction1.

On peut rajouter des options dans les fonctions line et plot, qui permettent notamment de fixer la fenêtre de visualisation ou la couleur des lignes. Utilisez les commandes line? et plot? pour afficher ces différentes options.

On peut additionner des graphiques. graphique1 + graphique2 contient à la fois les objets de graphique1 et ceux de graphique2.

Groupe E : Algorithme de seuil

Programmes à construire

On fixe un paramètre $c \in [0,4]$. Soit $f_c : [0,1] \to [0,1]$ la fonction définie par

$$f_c(x) = cx(1-x) \text{ pour tout } x \in [0,1].$$

L'intervalle $[0,1]$ est stable par f_c. Étant donnée une valeur $x_0 \in [0,1]$, on définit par récurrence la suite $(u_n)_{n\geq 0}$ telle que $u_0 = x_0$ et $u_{n+1} = f_c(u_n)$ pour tout $n \geq 0$.

Question 1 : Si on approche un nombre réel avec N *bits* de précision, combien de décimales exactes a-t-on, typiquement ? Quelle valeur de N choisir pour avoir 100 décimales de précision ?

Objectif 2 : Écrivez une fonction, appelée algorithme_de_seuil, qui prend en entrée un réel $x_0 \in [0,1]$ et un seuil $\varepsilon > 0$, et affiche :

- ▷ la plus petite valeur k telle que $|u_{k+1} - u_k| \leq \varepsilon$;
- ▷ la valeur de u_{k+1} correspondante.

Attention : si l'algorithme met trop de temps à trouver une telle valeur de k, il doit s'arrêter et afficher un message d'erreur.

Question 3 : Choisissez une valeur de c entre 0 et 3. Comment se comporte le nombre d'itérations k quand on diminue le seuil de précision ε ?

Étude de suites logistiques

À l'aide des programmes précédents, remplissez le tableau joint.

Une commande utile

Pour que les réels aient une précision de N *bits*, ajoutez au début de votre programme la ligne RealNumber = RealField(N). Il faudra toujours que la précision des réels soit meilleure que le seuil ε.