# Feuille de TP Python 03

Dynamique/ Graphique

## 1 Introduction

L'objet de ce TP est de présenter mathématiquement une gamme d'activités possibles autour de simulations de systèmes dynamiques avec illustrations graphiques pertinentes (ou pas).

Le système d'animation proposé dans les scripts est *très rudimentaire* mais, bon, ça marche avec le minimum possible de connaissances sur la bibliothèque graphique et on peut sauver les images créées dans un répertoire approprié pour post-traitement.

## 2 Suites récurrentes : escaliers ou escargots

**Exercice 1.—**On considère la fonction $f : x \mapsto \sqrt{3+x}$ (ou $f : x \mapsto \frac{1}{1+x^2}$) et étant donné $u_0 \in \mathbb{R}$, la suite $(u_n)_{n\in\mathbb{N}}$ satisfaisant la récurrence :

$$\forall n \in \mathbb{N},\ u_{n+1} = f(u_n)$$

**1.** Lire le code Python `esca-rgot-lier.py` capable de reproduire l'une ou l'autre de ces figures :

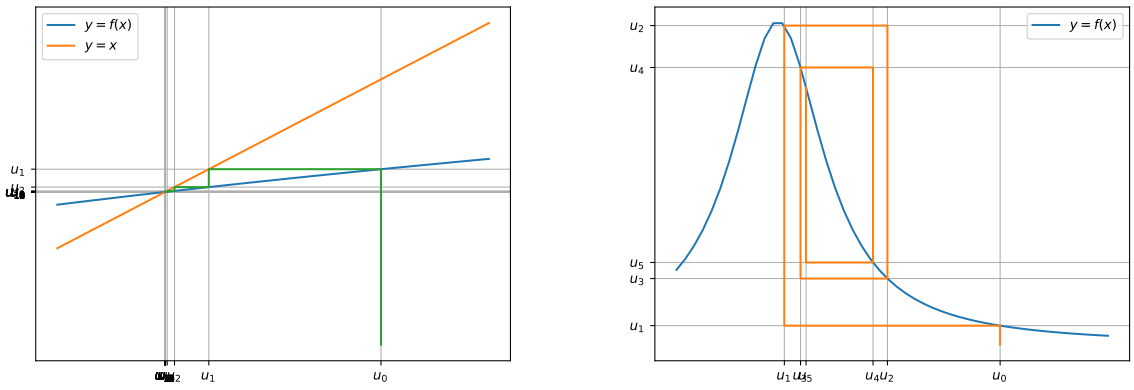



FIGURE 1 – Suite satisfaisant une récurrence $u_{n+1} = f(u_n)$

**2.** Corriger le code de sorte à écrire une fonction `RepGraphique(f,u0, nmax = 5, .... )` dont les premiers arguments sont la fonction $f$, le premier terme $u_0$ et qui sort le graphique standard précédemment présenté.

Tester en y mettant votre fonction de récurrence favorite.

**3.** Tester la fonction `AnimGraphique` (même entête, donnée en fin de script, à décommenter). Lire le code de cette fonction et comprendre où et comment l'animation est crée.

**4.** Ecrire une fonction `AnimGraphique2(f,u0, nmax = 5, ....)` en mixant les techniques de sorte à ce que :

- dans l'animation, on ne voie que 4 segments segments de l'escargot/escalier
- dans l'animation n'apparaissent que les abscisses/ordonnées des points dessinés. NB : Les méthodes `ax.get_xticks/ax.set_xtic` permettent de récupérer/fixer valeur/ les abscisses, les méthodes `ax.get_xticklabels/ax.set_xticklabels` permettent de récupérer/fixer valeur/ les labels correspondants. (Pour les ordonnées, changer les `x` en `y`.

**5.** Tester la fonction précécente pour comprendre ce qui se passe pour la suite récurrente $u^\lambda$ vérifiant $u_0^\lambda = 1$ et

$$\forall n \in \mathbb{N},\ u_{n+1}^\lambda = f^\lambda(u_n^\lambda)$$

où

$$f^\lambda(x) = \frac{\lambda}{1+x^2}$$

lorsque l'on fait varier $\lambda$ (entre 1 et 3).

# 3 La courbe du chien

On se propose maintenant d'utiliser la technique d'animation proposée précédement pour illustrer la problème de la courbe du chien :

— Le maître du chien marche le long du champ.

— Le chien, dans le champ, aperçoit son maître et commence à courir après lui.

— Sa vitesse [1] « au compteur » $v_c$ est légèrement supérieure à celle du maître $v_m$ ($v_m = \alpha.v_c$ où $\alpha < 1$).

— A chaque instant, le chien avance dans la direction de son maître.

Questions :

— Faire une simulation accompagnée d'une illustration graphique. Donner une estimation du temps mis par le chien pour rejoindre le maître (en proportion de $d/v_c$, c'est à dire le temps qu'il aurait mis si le maître ne bougeait que très très lentement).

— La courbe théorique [2] est donnée par l'équation

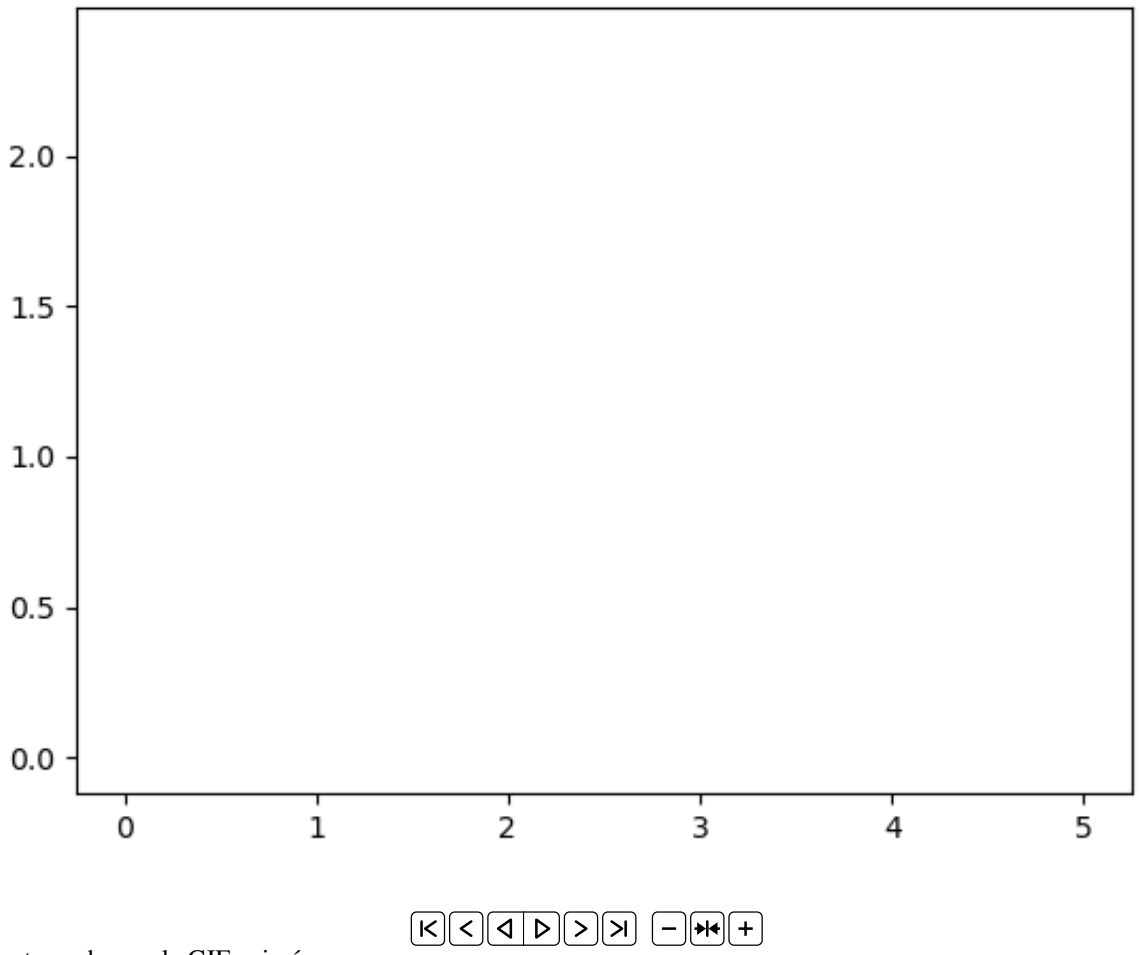
$$y = d.f(x/d)$$

où

$$f(t) = \frac{1}{2}\left(\frac{1}{\alpha-1}(1-t)^{1-\alpha} + \frac{1}{\alpha+1}(1-t)^{\alpha+1}\right) - \frac{\alpha}{\alpha^2-1}$$

Adapter votre simulation pour faire se superposer votre trajectoire et la trajectoire théorique.

— Reproduire l'animation suivante (visible sous Acrobat Reader)




Pour les autres, charger le GIF animé :
`http://jcleger.math.free.fr/maths/docs/happy-dog.gif`

1. On supose au début, pour tout simplifier que les vitesses sont constantes, que le maître va en ligne droite, on peut bien sûr changer tout cela ensuite
2. On suppose qu'à l'instant 0, le chien est en $(0,0)$, le maître en $(d,0)$ et marche à vitesse $v_m = \alpha.v_c$ le long de l'axe des ordonnées,

# 4 Psychédélisme

**Exercice 2.—**Dans cette partie on s'intéresse à l'action d'une diffusion simple sur une densité. Plus précisémment, on dispose d'une image numérique en niveaux de gris $M$ représentée par un tableau `M` bidimensionnel.

A chaque pas de temps, on lui fait subir la transformation suivante :

chaque pixel devient la moyenne entre lui-même et ses 4 voisins

Partant d'une image (par exemple) montrer par une animation, l'évolution de cette image. Reproduire l'animation suivante `http://jcleger.math.free.fr/maths/docs/happy-georgia.gif`

Indications :

Le code suivant permet le chargement, la transformation en matrice de niveaux de gris et l'affichage de l'image en gris.

```
M = plt.imread('diffusion.png')
M = M.mean(axis=2)
plt.imshow(M, cmap = 'gray')
```

On peut récupérer l'objet graphique par l'instruction

`img = plt.imshow(M)`

et récupérer/modifier le contenu du tableau de pixels par

`M = img.get_array()` / `M = img.set_array()`

Enfin, la technique d'animation, se fait toujours via une boucle dans laquelle on modifie les données des objets graphiques pour les redessiner en fion de boucle via

`fig.canvas.draw(); fig.canvas.flush_events()`

où `fig` est l'identitifant de figure, sauvegardé en début de procédure par

`fig, ax = plt.subplots()`

**Exercice 3.—**On reprend l'exercice précédent mais en ajoutant à chaque étape une « réaction intra-pixel ». Chaque pixel à un niveau de couleur R,G ou B que l'on peut récupérer dans le troisième canal de l'image

Par exemple, après chargement d'une image RGB par `plt.imread`, on peut assigner `R = M[:,:,0] ; G = M[:,:,1] ; B = M[:,:` ce qui permet de travailler directement avec les trois canaux.

On demande qu'à chaque étape chacun des trois canaux $R$, $G$ et $B$ se voit augmenté respectivement d'une quantité proportionnelle à : $R*(G-B)$, $G*(B-R)$, $B*(R-G)$.

Partant d'une image montrer par une animation, l'évolution de cette image par ce procédé.