TP 3 - Méthode d'Euler et équations différentielles autonomes : des exemples liés à la biologie

On note $N(t)$ le nombre d'individus d'une espèce donnée présents à l'instant $t \in \mathbb{R}_+$. La répartition spatiale des individus ne sera pas considérée ici. Les modèles considérés seront des équations différentielles ordinaires de la forme :

$$\begin{cases} N'(t) = f\left(N(t)\right), \quad t \in \mathbb{R}_+ \\ N(0) = N_0 \end{cases} \tag{E}$$

où N_0 représente le nombre d'individus à l'instant initial et f est une fonction de réaction qui sera spécifique à chaque modèle dans la suite.

On cherche à déterminer un modèle qui soit pertinent avec une situation réelle d'évolution de populations au cours du temps.

1 Approximation des solutions d'une équation différentielle : la méthode d'Euler

On se place sur un intervalle $[0, t_f]$ et on considère un problème de Cauchy d'ordre 1 ($N_0 \in \mathbb{R}$ fixé) :

$$\begin{cases} N'(t) &= f(N(t)) \\ N(0) &= N_0 \text{ (condition initiale)} \end{cases}$$

On subdivise l'intervalle $[0, t_f]$ en n sous-intervalles, de longueur $\mathrm{dt} = \frac{t_f}{n}$ et on note pour tout $k \in [\![0, n]\!]$, $t_k = k \times \mathrm{dt}$.
La **méthode d'Euler** consiste à approcher la dérivée par le taux d'accroissement :

$$N'(t_k) \approx \frac{N(t_{k+1}) - N(t_k)}{\mathrm{dt}}.$$

On obtient alors l'approximation suivante :

$$N(t_{k+1}) \approx N(t_k) + \mathrm{dt} \times f(N(t_k)).$$

On va noter y_k l'approximation de $N(t_k)$ pour tout $k \in [\![0, n]\!]$. La suite $(y_k)_{k \in [\![0,n]\!]}$ est alors définie par :

$$\begin{cases} y_{k+1} &= y_k + \mathrm{dt} \times f(y_k) \text{ pour } k \in [\![0, n-1]\!] \\ y_0 &= N_0 \end{cases}$$

1. Programmer une fonction `Euler(fonc,tf,N0,dt)` qui, étant donné un problème de Cauchy (E), donne une approximation de sa solution sur $[0, t_f]$, avec un pas de discrétisation `dt`. En sortie, on obtient donc une liste $[y_0, y_1, \ldots, y_n]$ qui représente les valeurs approchées de $[N(0), N(\mathrm{dt}), N(2\mathrm{dt}) \ldots, N(t_f)]$.

2 Le modèle de Malthus

Le modèle de Malthus est régi par l'équation différentielle :

$$\begin{cases} N'(t) &= aN(t) \\ N(0) &= N_0 \end{cases} \tag{1}$$

où $a \in \mathbb{R}_+^\star$ représente la différence entre le taux de natalité et le taux de mortalité.

2. Résoudre cette équation différentielle. Ce modèle vous semble-t-il pertinent ?
3. Simuler numériquement la solution de cette équation différentielle en utilisant la méthode d'Euler avec $a = 1.5$, $N_0 = 800$ et $t_f = 10$. Il faut pour cela définir la fonction `fMalthus` qui permet d'écrire l'équation différentielle (1) sous la forme de (E), puis appliquer la fonction `Euler` avec les paramètres donnés. On testera les pas dt = 1, dt = 0.1 et dt = 0.01.
4. Comparer ces approximations avec la solution exacte en traçant toutes les courbes représentatives sur le même graphe (on s'aidera d'une légende pour identifier les courbes).

Extrait de la notice du SCAV concernant le module matplotlib.pyplot

Matplotlib.pyplot

`import matplotlib.pyplot` as **plt**

`plt.plot(X,Y,'+-r')` ---- Génère la courbe des points définis par les listes `X` et `Y` (abscisses et ordonnées) avec les options :
- symbole : `'.'` point, `'o'` rond, `'h'` hexagone, `'+'` plus, `'x'` croix, `'*'` étoile, ...
- ligne : `'-'` trait plein, `'--'` pointillé, `'-.'` alterné, ...
- couleur : `'b'` **bleu** , `'r'` **rouge** , `'g'` **vert** , `'c'` **cyan** , `'m'` **magenta** , `'k'` **noir** , ...

`plt.bar(X,Y)` ---------- Génère l'histogramme des points définis par les listes `X` et `Y` (abscisses et ordonnées)

`plt.axis('equal')` ------ Rend le repère orthonormé

`plt.xlim(xmin,xmax)` ---- Fixe les bornes de l'axe des abscisses

`plt.ylim(ymin,ymax)` ---- Fixe les bornes de l'axe des ordonnées

`plt.show()` ------------ Affiche le graphique

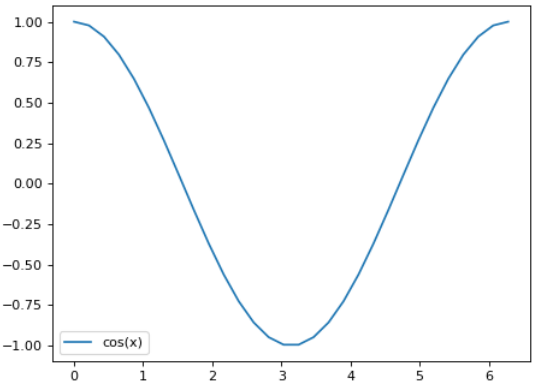
Exemple de mise en place d'une légende

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

x = np.linspace(0, 2*np.pi, 30)
y = np.cos(x)

plt.plot(x, y, label="cos(x)")
plt.legend()

plt.show()
```



3 Le modèle logistique

Le modèle logistique est régi par l'équation différentielle :

$$\left\{ \begin{array}{rcl} N'(t) & = & aN(t)(K - N(t)) \\ N(0) & = & N_0 \end{array} \right. \tag{2}$$

où $(a, K) \in (\mathbb{R}_+^\star)^2$ sont des constantes. Le paramètre K représente la capacité d'accueil du milieu et le paramètre a est lié aux taux de natalité et de mortalité.

5. Pour tout $t \in \mathbb{R}_+$, on suppose que $N(t) \neq 0$ et on pose $z(t) = \dfrac{1}{N(t)}$. Déterminer l'équation différentielle vérifiée par z.
6. Résoudre cette équation différentielle.
7. En déduire l'expression de N en fonction de t.
8. Déterminer la limite de N lorsque t tend vers $+\infty$ et justifier la dénomination de la constante K.
9. On fixe dans cette question $K = 1000$, $N_0 = 800$ et $a = 0.002$. On trace la courbe représentative de N au cours du temps. Ce modèle vous semble-t-il pertinent ?
10. Simuler numériquement la solution de cette équation différentielle (2) en utilisant la méthode d'Euler avec $t_f = 10$ et dt $= 0.01$. On pensera à définir la fonction `fLogistique` nécessaire pour appliquer la fonction `Euler`.
11. Comparer cette approximation avec la solution exacte.

4 Le modèle de Gompertz

Le modèle de Gompertz est régi par l'équation différentielle :

$$\left\{ \begin{array}{rcl} N'(t) & = & aN(t) \ln\left(\frac{K}{N(t)}\right) \\ N(0) & = & N_0 \end{array} \right. \tag{3}$$

où $(a, K) \in (\mathbb{R}_+^\star)^2$ sont des constantes. Le paramètre K représente la capacité d'accueil du milieu et le paramètre a est lié aux taux de natalité et de mortalité.

12. Pour tout $t \in \mathbb{R}_+$, on pose $z(t) = \ln\left(\frac{K}{N(t)}\right)$. Déterminer l'équation différentielle vérifiée par z.
13. Résoudre cette équation différentielle.
14. En déduire l'expression de N en fonction de t.
15. Déterminer la limite de N lorsque t tend vers $+\infty$ et justifier la dénomination de la constante K.
16. On fixe dans cette question $K = 1000$, $N_0 = 800$ et $a = 0.002$. On trace la courbe représentative de N au cours du temps pour un temps final de 10000. Ce modèle vous semble-t-il pertinent ?
17. Tester l'approximation numérique de cette solution de l'équation différentielle (3) par la méthode d'Euler avec $t_f = 10$ et dt $= 0.01$. On pensera à définir la fonction `fGomperz` nécessaire pour appliquer la fonction `Euler`.

5 Approximation des solutions d'une équation différentielle : la méthode de Heun ou de Runge-Kutta d'ordre 2

On se place sur un intervalle $[0, t_f]$ et on considère un problème de Cauchy d'ordre 1 ($N_0 \in \mathbb{R}$ fixé) :

$$\begin{cases} N'(t) &= f(N(t)) \\ N(0) &= N_0 \text{ (condition initiale)} \end{cases}$$

On subdivise l'intervalle $[0, t_f]$ en n sous-intervalles, de longueur $\text{dt} = \frac{t_f}{n}$ et on note pour tout $k \in [\![0, n]\!]$, $t_k = k \times \text{dt}$.
La **méthode de Heun** consiste à poser :

$$\begin{cases} k_1 &= \text{dt } f(N(t_k)) \\ k_2 &= \text{dt } f(N(t_k) + k1) \\ N(t_{k+1}) &\approx N(t_k) + \dfrac{k_1 + k_2}{2} \end{cases}$$

18. Programmer une fonction `Heun(fonc,tf,N0,dt)` qui étant donné un problème de Cauchy (E) donne une approximation de sa solution sur $[0, t_f]$, avec un pas de discrétisation `dt`.
19. Tester cette fonction sur l'équation logistique avec les mêmes paramètres que précédemment.
20. Comparer les résultats pour $\text{dt} = 0.1$ avec les méthodes d'Euler et de Heun.
21. Programmer une fonction `Erreur(dt)` qui calcule le maximum de l'écart entre la solution exacte et la solution approchée pour les méthodes d'Euler et de Heun à dt fixé.
22. Tracer l'évolution de l'erreur en fonction de dt pour les deux méthodes. Laquelle est la plus précise ? On prendra `dt=[10**(-k) for k in np.linspace(1,5.5,15)]` après avoir importé `import numpy as np`.
23. Le graphe ne montre pas bien ce qui se passe pour dt petit, comment pourrait-on changer d'échelle ?
24. **Bonus :** Tracer l'évolution du logarithme de l'erreur en fonction du logarithme de dt. Vous devez trouver des droites pour les deux méthodes. À l'aide de la fonction `linregress` (régression linéaire) du module `scipy.stats`, déterminer les pentes de ces droites. Qu'en déduisez-vous sur l'ordre de grandeur de l'erreur pour ces deux méthodes ?

Exemple de régression linéaire avec Python

* Regression linéaire :

- on peut aussi faire `lr = scipy.stats.linregress([3, 4, 6, 8], [6.5, 4.2, 11.8, 15.7])`.
- renvoie un tuple avec 5 valeurs (ici, `(2.1796610169491526, -1.8932203389830509,0.93122025491258043, 0.068779745087419575, 0.6032088854571009`
  - la pente.
  - l'ordonnée à l'origine.
  - le coefficient de corrélation, positif ou négatif (pour avoir le coefficient de détermination R2, prendre le carré de cette valeur).
  - la p-value.
  - l'erreur standard de l'estimation du gradient.