Chapitre # (NUM) 2 Méthodes numériques pour les équations différentielles d'ordre 1

Rien ne se passe dans l'Univers sans qu'un minimum ou un maximum apparaisse.

— Leonhard Euler

Résumé & Plan

L'objectif de ce chapitre est de savoir mettre en place la méthode d'Euler afin de savoir résoudre des équations différentielles de manière approchée.

- Les chapitres d'Informatique sont composés de cours et d'exercices intégrés. Le cours sera projeté au tableau.
- Il n'est pas attendu que toute la classe aborde tous les exercices. Traitez donc en priorité les exercices présents dans la liste donnée à chaque début de séance.
- Exercices ◐ / **Pour aller plus loin** : exercices plus difficiles, ou plus techniques. À ne regarder que si les autres sont bien compris.

1. ORDRE 1 DANS $\mathbb{R}$

1.1. Aspects mathématiques

On rappelle que l'on appelle « problème de Cauchy d'ordre 1 » tout problème de la forme :

$$\begin{cases} y' &= f(t,y), \\ y(t_0) &= y_0, \end{cases} \quad \text{avec} \quad f : \mathrm{I} \times \mathbb{R} \longrightarrow \mathbb{R} \quad \text{continue},$$

où I est un intervalle et $y_0 \in \mathbb{R}$. On rappelle de plus que :

- y est la fonction inconnue : on cherche à détermine la valeur de $y(t)$ pour $t \in \mathrm{I}$, l'intervalle de définition de y,
- f est l'expression qui apparaît dans l'équation différentielle faisant intervenir tout sauf y', elle peut utiliser t et $y(t)$ (d'où une fonction de 2 variables).
- $y(t_0) = y_0$ est la condition initiale, garantissant l'unicité de la solution.

Exemple 1 **(Équations différentielles linéaires)** Dans chaque exemple, préciser la fonction f, l'intervalle de définition I, ainsi que l'unique solution.

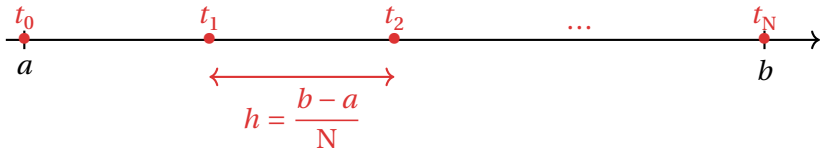
- $y'(t) = t$, $y(0) = 0$

- $y'(t) = y(t)^2$, $y(0) = 1$

- $y'(t) - y(t) = \ln(t)$, $y(1) = -1$

Dans ce TP, nous ne considèrerons que des intervalles bornés de la forme $\mathrm{I} = [0, \tau]$ où $\tau > 0$. L'ensemble des fonctions s'adaptent facilement à un intervalle ne démarrant pas forcément en 0.

Discrétisation d'un segment $[a, b]$. On considère ici une suite $(t_i)_{i=0}^{\mathrm{N}}$ découpant l'intervalle $[a, b]$ de manière régulière en $\mathrm{N}+1$ points (ou N sous-intervalles). Ici N est un entier positif.



De manière explicite, on a :

$$\boxed{\forall i \in [\![0\,,\mathrm{N}]\!], \quad t_i = a + i\frac{b-a}{\mathrm{N}} = a + ih.}$$

En effet,

- c'est une suite arithmétique de raison h, de premier terme $t_0 = a$. Donc :
  $\forall i \in [\![0\,,\mathrm{N}]\!], \quad t_i = a + ih.$
- Mais on connait aussi la valeur terminale $t_{\mathrm{N}} = b$, ce qui donne en faisant $i = \mathrm{N}$ dans l'égalité précédente : $a + \mathrm{N}h = b \iff h = \dfrac{b-a}{n}$.
- On a donc montré : $\forall i \in [\![0\,,\mathrm{N}]\!], \quad t_i = a + i\dfrac{b-a}{\mathrm{N}} = a + ih.$

En particulier, si $a = 0, b = \tau$, on obtient :

$$\forall i \in [\![0\,,\mathrm{N}]\!], \quad t_i = i\frac{\tau}{\mathrm{N}} = ih.$$

Valeurs approchées de la solution exacte *via* la méthode d'Euler. La méthode est basée sur une *heuristique* ou une « analyse / synthèse ».

Si y est une solution *(qu'on ne sait a priori pas calculer puisqu'on utilise une méthode numérique)* **alors** on peut déduire une relation de récurrence sur la suite $(y(t_i))_{i=0}^{\mathrm{N}}$, c'est-à-dire sur les valeurs de y aux points de la subdivision. Cette relation de récurrence permet alors de calculer explicitement $y(t_1), \dots, y(t_{\mathrm{N}})$ à partir de $y(t_0) = y_0$ qui est donné par la condition initiale.

Notation pour les valeurs d'une solution aux points de la subdivision

Afin d'alléger la présentation, on notera $x_i, y_i, z_i, \dots$ les valeurs de x, y, z aux points t_i, au lieu de $x(t_i), y(t_i), z(t_i), \dots$.

Mais comment former une relation de récurrence sur les y_i ? On utilise la définition du nombre dérivé. En effet, en évaluant l'équation différentielle en $t = t_i$,

$$\forall t \in \mathrm{I}, \quad y'(t) = f(t, y(t)) \implies \forall i \in [\![0\,,\mathrm{N}]\!], \quad y'(t_i) = f(t_i, y(t_i)).$$

Or, pour h assez petit (tel que $t_i + h \in \mathrm{I}$), on a :

$$y'(t_i) \approx \frac{y(t_i+h) - y(t_i)}{h} \iff y(t_i + h) \approx y(t_i) + hy'(t_i).$$

En combinant ces deux relations, on déduit :

$$\boxed{\forall i \in [\![0\,,\mathrm{N}-1]\!], \quad y_{i+1} \approx y_i + hf(t_i, y_i)}.$$

Avant de la mettre en place en Python, récapitulons le principe.

Méthode d'Euler

1. **[Subdivision]** On commence par subdiviser l'intervalle $[0, \tau]$ de manière uniforme à l'aide de $\mathrm{N}+1$ points espacés d'un pas de $h = \dfrac{\tau}{\mathrm{N}}$. Plus précisément, on pose :
   $$t_0 = 0, \quad t_1 = h, \quad t_2 = 2h, \quad \dots, \quad t_i = ih, \quad \dots, \quad t_{\mathrm{N}} = \boxed{\mathrm{N}h = \tau.}$$
   La relation encadrée est à bien garder en mémoire.
2. **[Heuristique]** On considère que pour h petit :
   si y est une solution, **alors :** $\forall t \in \mathrm{I}, y'(t) = f(t, y(t)) \approx \dfrac{y(t+h) - y(t)}{h}$.
3. **[Construction d'une suite récurrente]** La suite de points $(y_i)_{i \in i \in [\![0,\mathrm{N}]\!]}$ satisfaisant :
   $$\forall i \in [\![0\,,\mathrm{N}-1]\!], \quad y_{i+1} = y_i + h \times f(t_i, y_i)$$
   sera alors une bonne approximation de la solution inconnue aux points de la subdivision. La condition initiale que l'on souhaite pour $y(0)$ fixe alors $y(0)$.

Remarque 1 On pourrait alors dans un second temps essayer quantifier l'erreur d'approximation : quel pas h ou nombre de points $\mathrm{N}+1$ choisir ? qu'appelle-t-on erreur d'approximation ? *etc.* Nous ne répondrons en revanche pas à ces problèmes d'analyse qualitative, qui sont hors-programme.

1.2. Aspects informatiques

On doit stocker étape par étape les termes de la suite $(y_i)_i$ puis ensuite tracer cette suite sur un graphique, on peut donc utiliser naturellement une liste. La discrétisa-

tion peut s'obtenir quant à elle facilement avec la commande `np.linspace` rencontrée dans le Chapitre (NUM) 1.

(Méthode d'EULER)

```
import numpy as np
def euler(f, y0, tau, N):
    """
    f : fonction, y0 : valeur en zéro,
    tau : borne max de l'intervalle,
    N : nombre d'intervalles de la subdivision->retourne le couple
    (subdivision, solution approchée) selon la méthode d Euler
    """
    h = tau/N
    T = np.linspace(0, tau, N+1) # N+1 points dans la discrétisation
    Y = [y0]
    for i in range(N):
        y = Y[-1] + h*f(T[i], Y[-1])
        Y.append(y) # Y[-1] est le dernier y ajouté
    return T, Y
```

Exemple 2 (Fonction exponentielle) Testons avec la fonction exponentielle, *i.e.* l'équation différentielle $y' = y$, qui est :

$$\forall t \in \mathbb{R}, \quad y'(t) = y(t) = f(t, y(t))$$

avec :

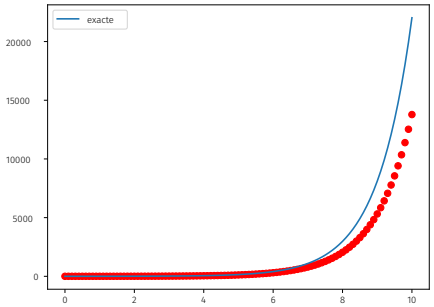
$$\forall (t,x) \in \mathbb{R} \times \mathbb{R}, \quad f(t,x) = x.$$

On commence donc ensuite par définir la fonction f en Python, puis on applique la fonction précédente.

```
def f(t,x):
    return x
y0, tau, N = 1, 10, 100
T, Y = euler(f, y0, tau, N)
```

On peut ensuite passer au tracé.

```
plt.plot(T, Y, 'ro') # sol. approchée (points en rouge non reliés, label="approchee")
plt.plot(T, [np.exp(x) for x in T], label="exacte") # solution exacte
plt.legend()
```



La méthode vous semble-t-elle performante ? On peut par exemple comparer la valeur terminale en $t = \tau = 10$ avec la vraie valeur $\exp(10)$.

```
>>> abs(Y[-1]-ma.exp(10)) # pas terrible
8245.853454984443
```

On peut ensuite augmenter un peu le nombre de points, c'est un peux mieux.

```
y0, tau, N = 1, 10, 10000
T, Y = euler(f, y0, tau, N)
>>> abs(Y[-1]-ma.exp(10)) # c'est mieux
109.78445572831333
```

En dernier test, analysons la valeur approchée de la constante de NÉPER $e = \exp(1)$ donnée par la méthode d'EULER et comparons avec la valeur exacte.

```
>>> i = int(N/tau) # l'entier tel que ti = 1 (avec int on enlève le .0 à la fin)
>>> abs(Y[i]-ma.exp(1)) # pas mal ici (proche du point initial)
0.0013578962231490799
```

Bien sûr, plus on augmente le nombre de points plus cela prend de temps côté ordinateur, on ne peut pas tout avoir.

1.3. Utilisation

Exercice 1 | Équation différentielle non linéaire (1) Solution Créer une fonction d'en-tête `trace_approx1()` qui trace les approximations de l'équation différentielle $y'(t) = t^3 - y(t)^3$ sur l'intervalle $[0,2]$ avec différentes conditions initiales : $y_0 = -3, y_0 = -2, y_0 = 0, y_0 = 4$. *On commencera par créer une liste comportant ces conditions initiales, que l'on parcourra à l'aide d'une boucle* **for**.

Exercice 2 | Équation différentielle non linéaire (2) Solution Créer une fonction d'en-tête `trace_approx2()` qui trace l'approximation de $y' = \sin(ty)$. *On se placera sur l'intervalle* $[0,5]$ *et on choisira* $y_0 = 12$ *et* $N = 1000$.

Exercice 3 | Dynamique malthusienne d'une population Solution En l'absence de prédateurs et avec une nourriture abondante, l'évolution d'une famille de lapins, par exemple, suit un modèle de MALTHUS :

$$\begin{cases} y' &= ry, \\ y(0) &= 2, \end{cases} \quad \text{où } r \in \mathbb{R}.$$

Le signe du coefficient r nous dit si la mortalité est supérieure à la natalité au sein de la population (ou vice-versa).

1. Résoudre cette équation différentielle.

2. On souhaite maintenant obtenir la représentation graphique de cette fonction. Tracer la courbe solution à l'aide de Python.
3. Dans la même fenêtre graphique, représenter la solution du système de MALTHUS pour $r_1 = \ln(2)$ et $r_2 = 1$. La première courbe devra être représentée en rouge et la seconde en bleu. On effectuera le tracé sur l'intervalle $[0,10]$.
4. Résoudre cette équation différentielle à l'aide de la méthode d'EULER, en commençant par créer la fonction `f` associée à cette équation différentielle. On se placera sur l'intervalle $[0,10]$ ($\tau = 10$) avec $r = -\ln(2)$ par exemple, et on tracera sur le même graphique la solution obtenue avec une subdivision en 10 points, en 100 points et en 1000 points, avec légende.

Exercice 4 | Chute libre d'un mobile Solution L'équation différentielle sur la vitesse v d'un mobile de masse m en chute libre sans vitesse initiale, soumis à la pesanteur g, et de frottements proportionnels au carré de la vitesse est la suivante :

$$g - \frac{k}{m}v^2 = v'.$$

Cette équation différentielle correspond à un mobile soumis au poids, à des frottements supposés « quadratiques » c'est-à-dire proportionnels à l'opposé du carré de la vitesse (le coefficient de proportionnalité est k). Elle s'obtient à l'aide de la relation fondamentale de la dynamique (RFD) projetée dans un repère.

1. À l'aide de la méthode d'EULER, proposer un tracé de v pour $m = \frac{1}{g}, k = 1$ sur l'intervalle $[0,1]$, de pas $0.1, 0.05, 0.01, 0.005$. *On rappelle que* $g \approx 9.81 \mathrm{N/kg}$ *sur la Terre, et on commencera par créer une liste contenant les différents pas à tester.*
2. Interpréter l'allure des courbes obtenues. Que se passe-t-il lorsque $t \xrightarrow{t\to\infty} \infty$?
3. Donner par lecture graphique une valeur approchée de la vitesse à l'instant 0.3.

2. ORDRE 1 DANS $\mathbb{R}^2$ SUR UN EXEMPLE

Considérons un système d'équations différentielles de la forme :

$$\begin{cases} y' &= f(t,y,z), \\ z' &= g(t,y,z), \\ y(t_0) &= y_0 \\ z(t_0) &= z_0, \end{cases} \quad \text{avec} \quad f : \mathrm{I} \times \mathbb{R} \longrightarrow \mathbb{R}, g : \mathrm{I} \times \mathbb{R} \longrightarrow \mathbb{R} \quad \text{continues},$$

où I est un intervalle et $y_0, z_0 \in \mathbb{R}$. Les fonctions inconnues sont maintenant y et z. Alors la méthode employée précédemment s'applique encore, il suffit d'approcher la dérivée de y et de z à l'aide du taux d'accroissement, comme nous l'avons fait précédemment. Nous ne présentons pas de fonction générale ici (étendant `euler`), on travaille uniquement sur un exemple.

MODÈLE DE LOKTA-VOLTERRA. Si deux espèces dont les populations sont représentées par y et z se partagent le milieu, il est naturel d'envisager une prédation entre ces deux espèces.

- On appellera dans la suite *proie* la population de cardinal $y(t)$ au temps t,
- et prédateur celle de cardinal $z(t)$.

Pour tenir comptes des interactions, on considère le système d'équations différentielles ci-après :

$$y'(t) = \underbrace{(a - bz(t))}_{\text{Taux de d'évolution variable incluant les prédateurs}} y(t),$$

(une augmentation de z freinera la population y : prédation de z sur y)

$$z'(t) = \underbrace{(-c + dy(t))}_{\text{Taux d'évolution variable incluant les proies (nourriture)}} z(t), \quad a,b,c,d > 0.$$

(une augmentation de y favorisera la population z : z sert de nourriture à y)

Résolution approchée par la méthode d'Euler. Puisque le système précédent est clairement non linéaire, on envisage une résolution purement numérique.

Exercice 5 | Méthode numérique pour Lokta–Volterra Solution On discrétise un intervalle $[0,\tau]$ comme nous l'avons fait précédemment, on note toujours h le pas et $N+1$ le nombre de points de la discrétisation.

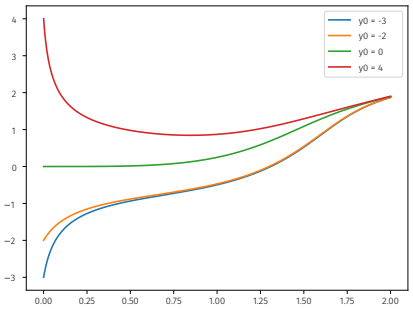
1. Justifier l'introduction des suites récurrentes suivantes pour $i \in [\![0, N-1]\!]$,
$$\begin{cases} y_{i+1} &= y_i + h(ay_i - bz_iy_i), \\ z_{i+1} &= z_i + h(-cz_i + dy_iz_i). \end{cases}$$
2. Écrire une fonction `euler_LoktaVolt` de paramètres `y0, z0, tau, N` qui retourne le tableau de discrétisation pour le temps, une liste pour les proies et pour les prédateurs. S'en servir pour tracer une solution approchée pour les proies et les prédateurs sur $[0,\tau]$. *Les constantes `a, b, c, d` pourront être mises en variables globales avant le début de la fonction.*
3. On pourra faire prendre aux paramètres les valeurs ci-après, et tracer sur $[0,12]$:
   - `a, b, c, d = 2, 6, 2, 6,`
   - puis `a, b, c, d = 2, 12, 2, 12`.

3. SOLUTIONS DES EXERCICES

Solution (exercice 1) Énoncé

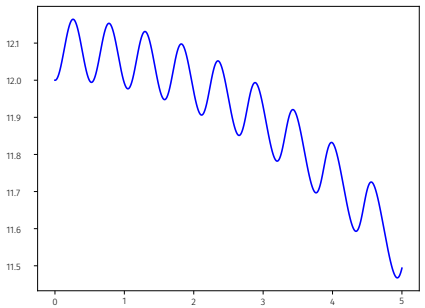
```
def f(t, x):
    return t**3-x**3

def trace_approx1():
    cond_init = [-3, -2, 0, 4]
    for y0 in cond_init:
        T, Y = euler(f, y0, 2, 10**3)
        plt.plot(T, Y, label="y0 = "+str(y0))
        plt.legend()
trace_approx1()
```

```
plt.show()
```



Solution (exercice 2) Énoncé

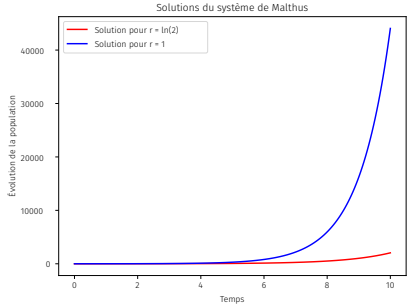
```
def f(t, x):
    return ma.sin(t*x)
y0, tau, N = 12, 5, 10**3
T, Y = euler(f, y0, tau, N)
plt.plot(T, Y, color = "blue")
```



Solution (exercice 3) Énoncé

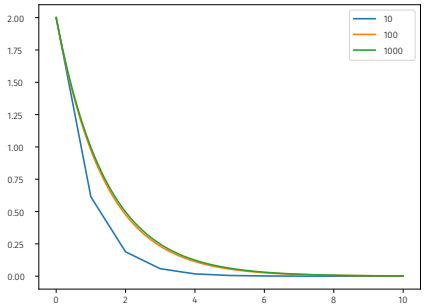
1. On se place sur $I = [0, +\infty[$. Il s'agit d'une équation homogène dont la solution est : $y(t) = Ce^{rt}$ où $C \in \mathbb{R}$, définie pour tout $t \in I$. La condition initiale impose que : $y(0) = 2 = C$. Ce problème de CAUCHY admet donc pour unique solution la fonction $y : I \longrightarrow \mathbb{R}$ définie par : $\forall t \in I,\ y(t) = 2e^{rt}$

2.
```
import numpy as np
r1 = np.log(2)
r2 = 1
X = np.linspace(0, 10, 10**3)
Y1 = [2*np.exp(r1*x) for x in X]
Y2 = [2*np.exp(r2*x) for x in X]
plt.plot(X, Y1, color = "red", label = "Solution pour r = ln(2)")
plt.plot(X, Y2, color = "blue", label = "Solution pour r = 1")
plt.xlabel("Temps")
plt.ylabel("Évolution de la population")
plt.title("Solutions du système de Malthus")
plt.legend()
```



3. On utilise là encore la fonction `euler` créée au début.

```
r = -np.log(2)
def f(t, x):
    return r*x

y0, tau = 2, 10
for n in [10, 100, 1000]:
    T, Y = euler(f, y0, \
    ↪ tau, n)
    plt.plot(T, Y, label = \
    ↪ n)
plt.legend()
```



Solution (exercice 4) Énoncé

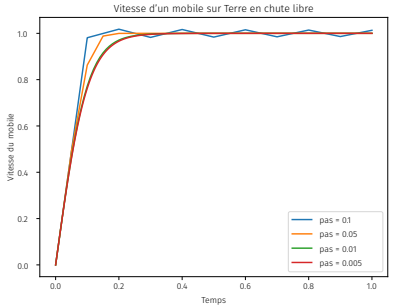
1. On utilise là encore la fonction `euler` créée au début. Là il convient tout d'abord d'exhiber clairement la fonction f. On a :

$$\forall t \in \mathbb{R}^+, \quad g - \frac{k}{m} v(t)^2 = v'(t) \iff v'(t) = -\frac{k}{m} v(t)^2 + g = f(t, v(t)),$$

avec :

$$\forall (t,x) \in \mathbb{R}^+ \times \mathbb{R}, \quad f(t,x) = x.$$

```
g, k = 9.81, 1
m = 1/g
def f(t, x):
    return -k/m*x**2+g

y0, tau = 0, 1 # pas de vitesse initiale
for h in [0.1, 0.05, 0.01, 0.005]:
    N = int(tau/h)
    T, Y = euler(f, y0, tau, N)
    plt.plot(T, Y, label = "pas = "+str(h))

plt.xlabel("Temps")
plt.ylabel("Vitesse du mobile")
plt.title("Vitesse d'un mobile sur Terre en chute libre")
plt.legend()
```

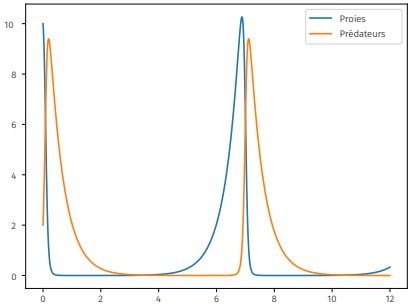


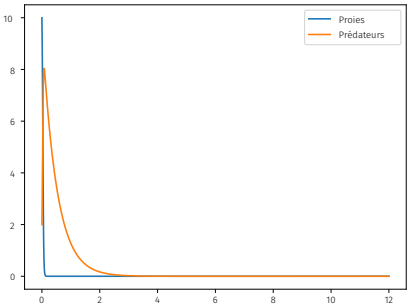
2. On constate une vitesse en racine carrée du temps. La vitesse limite est ici 1, on peut montrer de manière générale que la limite est forcément un point d'équilibre du système c'est-à-dire une solution constante, donc ici $\pm\frac{mg}{k} = \pm 1$ avec les valeurs des constantes données dans l'énoncé.

3. On obtient $v(t = 0.3) \approx 1$. On pourrait aussi le trouver plus précisément en exploitant l'une des listes `Y` données par la méthode d'EULER.

Solution (exercice 5) Énoncé

1. On approche classiquement $y'(t_i) \approx \frac{y(t_{i+1}) - y(t_i)}{h}$ pour tout i, même chose pour y. On obtient alors le système récurrent proposé.

2.
```
import matplotlib.pyplot as plt
import numpy as np

def f1(t, y, z):
    return (a-b*z)*y
def f2(t, y, z):
    return (-c+d*y)*z

def euler_LoktaVolt(y0, z0, tau, N):
    h = tau/N
    T = np.linspace(0, tau, N+1)
    Y = [y0]
    Z = [z0]
    for k in range(N):
        Y.append(Y[-1] + h*f1(T[k], Y[-1], Z[-1]))
        Z.append(Z[-1] + h*f2(T[k], Y[-1], Z[-1]))
    return T, Y, Z
```
Voici ce que l'on obtient pour un premier jeu de paramètres. On trace les solutions sur $[0,20]$.
```
N = 10**3
tau = 12
a, b, c, d = 2, 2, 2, 2
y0, z0 = 10, 2
T, Y, Z = euler_LoktaVolt(y0, z0, tau, N)
plt.plot(T, Y, label = 'Proies')
plt.plot(T, Z, label = 'Prédateurs')
plt.legend()
```


3. Dans le cas précédent, avec des taux de reproduction relativement équilibrés les deux familles reviennent dans le même état en permanence : les fonctions présentent des oscillations. Avec des données initiales différentes (plus de prédateurs initialement cette fois) on reste dans le même type d'évolution, voir ci-dessous.
```
a, b, c, d = 2, 6, 2, 6
y0, z0 = 10, 2
T, Y, Z = euler_LoktaVolt(y0, z0, tau, N)
plt.plot(T, Y, label = 'Proies')
plt.plot(T, Z, label = 'Prédateurs')
plt.legend()
```


En revanche si l'on augmente à la fois le taux de prédation et le taux de reproduction des prédateurs en fonction du nombre de proies mangées, on constate une extinction relativement rapide de la famille des proies.
```
a, b, c, d = 2, 12, 2, 12
y0, z0 = 10, 2
T, Y, Z = euler_LoktaVolt(y0, z0, tau, N)
plt.plot(T, Y, label = 'Proies')
plt.plot(T, Z, label = 'Prédateurs')
plt.legend()
```