Correction TP3 - 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. On propose la fonction suivante :

```
from math import *
def Euler(fonc,tf,N0,dt):
    t=[0]
    N=[N0]
    while t[-1]+dt<=tf:
        x=N[-1]+fonc(N[-1])*dt
        N.append(x)
        t.append(t[-1]+dt)
    return(t,N)
```

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. On reconnaît une équation différentielle linéaire du premier ordre à coefficient constant, de solution :

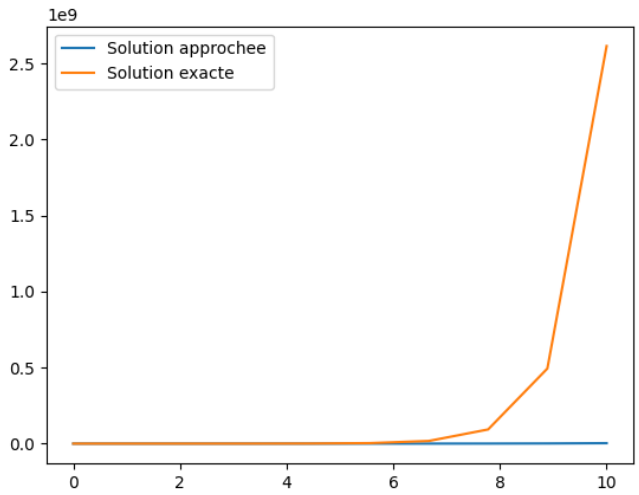
$$N(t) = N_0 e^{at}.$$

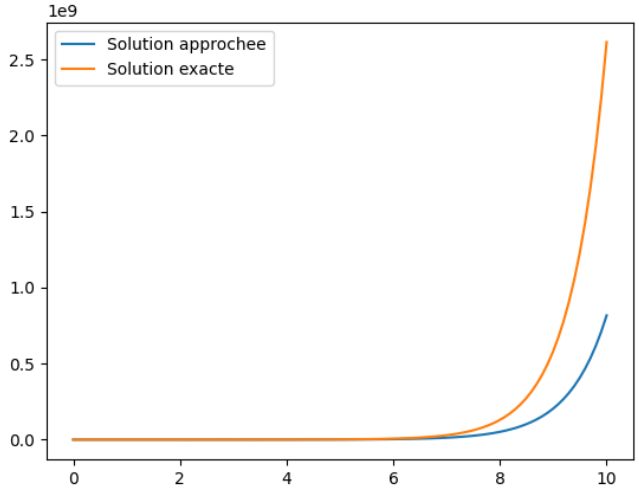
Il y a explosion exponentiellemenent rapide du nombre d'individus de l'espèce, cela n'est pas cohérent avec le nombre limité de ressources d'un milieu réel.

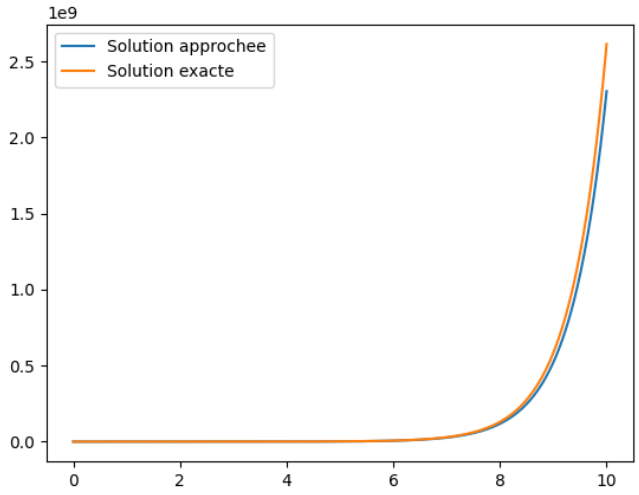
3. Simuler numériquement la solution de cette équation différentielle en utilisant la méthode d'Euler avec $a = 1.5$, $N_0 = 800$ et $t_f = 10$. On testera les pas dt $= 1$, dt $= 0.1$ et dt $= 0.01$.

```
import matplotlib.pyplot as plt
def fMalthus(N):
    return(1.5*N)

def SolutionMalthus(t,N0):
    return(N0*exp(1.5*t))

tf=10
dt=0.01
N0=800

T,Approx=Euler(fMalthus,tf,N0,dt)
plt.plot(T,Approx,label="Solution approchee")

Vraie=[SolutionMalthus(t,N0) for t in T]
plt.plot(T,Vraie,label="Solution exacte")
plt.legend()
plt.show()
```

On obtient les graphiques suivants pour les dt demandés, dans l'ordre :







4. On remarque qu'il faut un dt petit pour que l'approximation coïncide avec la solution exacte.

3 Le modèle logistique

Le modèle logistique est régi par l'équation différentielle :

$$\begin{cases} N'(t) &= aN(t)(K - N(t)) \\ N(0) &= N_0 \end{cases} \tag{2}$$

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. Soit $t \in \mathbb{R}_+$, on pose $z(t) = \dfrac{1}{N(t)}$, z est dérivable comme quotient d'une fonction qui l'est, avec

un dénominateur qui ne s'annule pas. On a :

$$\begin{aligned} z'(t) &= -\frac{N'(t)}{N^2(t)} \\ &= -\frac{aN(t)(K-N(t))}{N^2(t)} \\ &= -\frac{a(K-N(t))}{N(t)} \\ &= -aK\frac{1}{N(t)} + a \\ &= -aKz(t) + a \end{aligned}$$

donc

$$\forall t \in \mathbb{R}_+,\ \boxed{z'(t) = -aKz(t) + a}.$$

6. On reconnaît une équation différentielle du premier ordre à coefficients constants. L'équation homogène associée est :

$$\forall t \in \mathbb{R}_+,\ z'(t) = -aKz(t).$$

Ses solutions sont les $\{z_h : t \mapsto Ce^{-aKt},\ C \in \mathbb{R}\}$.

On cherche une solution particulière constante, on trouve $z_p(t) = \frac{1}{K}$.

Ainsi, par théorème fondamental, on obtient :

$$\forall t \in \mathbb{R}_+,\ z(t) = Ce^{-aKt} + \frac{1}{K}.$$

Il reste à déterminer C en sachant que $z(0) = \frac{1}{N_0}$. On obtient $C = \frac{K-N_0}{KN_0}$. Finalement, on a :

$$\forall t \in \mathbb{R}_+,\ \boxed{z(t) = \frac{K-N_0}{KN_0}e^{-aKt} + \frac{1}{K}}.$$

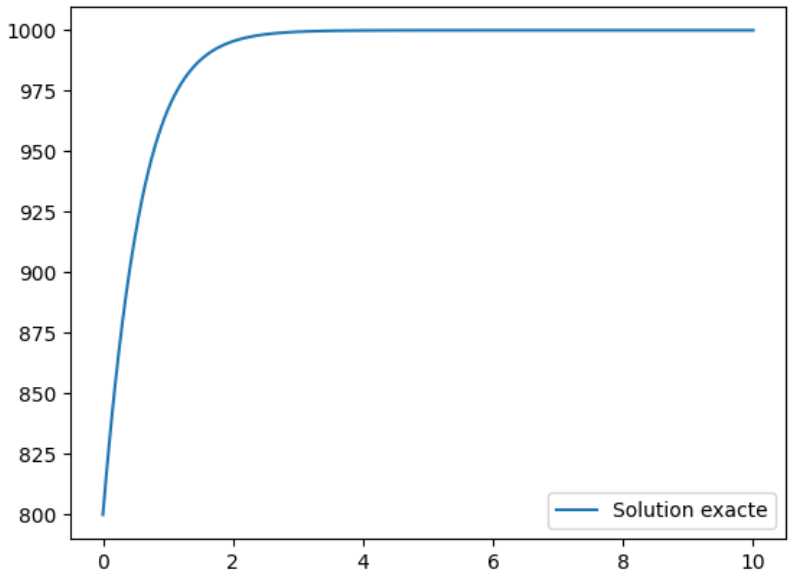
7. Comme $\forall t \in \mathbb{R}_+,\ N(t) = \frac{1}{z(t)}$ on en déduit :

$$\forall t \in \mathbb{R}_+,\ \boxed{N(t) = \frac{KN_0}{(K-N_0)e^{-aKt} + N_0}}.$$

8. On remarque que $\lim\limits_{t\to+\infty} N(t) = K$ donc K représente la capacité d'accueil du milieu.
9. On trace la courbe grâce au programme suivant :

```
def SolutionLogistique(t,N0):
    a=0.002
    K=1000
    return(K*N0/((K-N0)*exp(-a*K*t)+N0))
Vraie=[SolutionLogistique(t,N0) for t in T]
plt.plot(T,Vraie,label="Solution exacte")
plt.legend()
plt.show()
```

On obtient le graphique suivant :

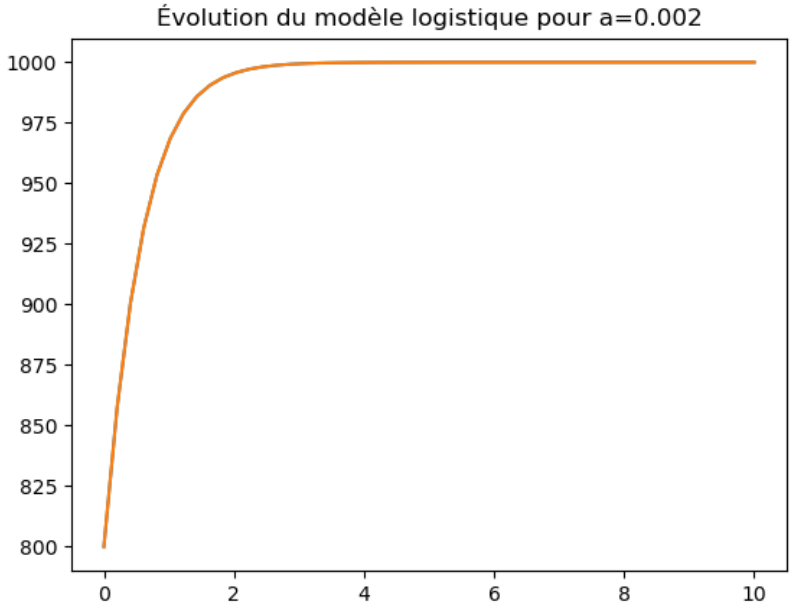


Le nombre d'individus semble atteindre un seuil à K, cela est cohérent avec un environnement très stable.

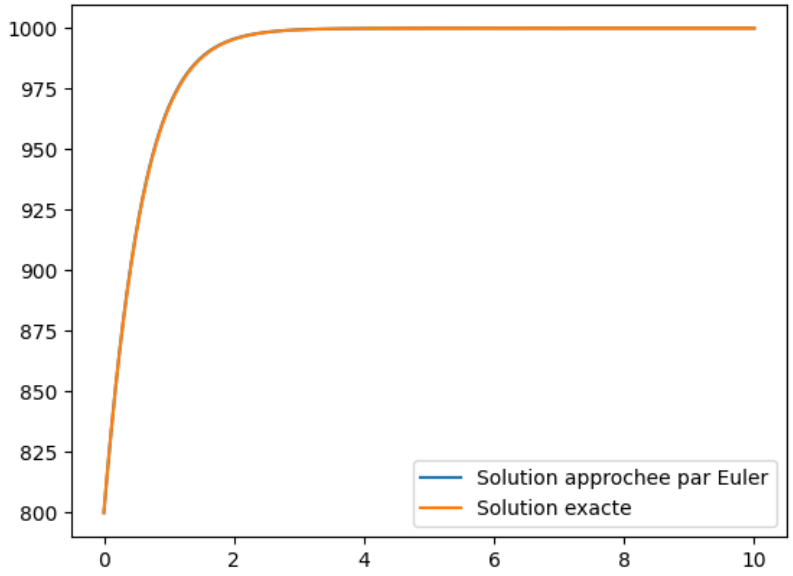
10. On simule grâce au code suivant :

```
def fLogistique(N):
    a=0.002
    K=1000
    return(a*N*(K-N))

tf=10
dt=0.01
N0=800

T,Approx=Euler(fLogistique,tf,N0,dt)
plt.plot(T,Approx)
plt.title('Évolution du modèle logistique pour a=0.002')
plt.show()
```

On obtient la courbe suivante :



11. On obtient les courbes suivantes :



Les courbes sont quasi-indiscernables, l'approximation est bonne.

4 Le modèle de Gompertz

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

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

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. Soit $t \in \mathbb{R}_+$, on pose $z(t) = \ln\left(\frac{K}{N(t)}\right)$, z est dérivable comme composée et quotient de fonctions qui le sont, avec un dénominateur qui ne s'annule pas. On a :

$$\begin{aligned} z'(t) &= -\frac{KN'(t)}{N^2(t)} \times \frac{N(t)}{K} \\ &= -\frac{N'(t)}{N(t)} \\ &= -a \ln\left(\frac{K}{N(t)}\right) \\ &= -az(t) \end{aligned}$$

donc on a montré :

$$\forall t \in \mathbb{R}_+, \quad \boxed{z'(t) = -az(t)}.$$

13. Les solutions sont de la forme :

$$t \mapsto Ce^{-at}, C \in \mathbb{R}.$$

On remarque que $C = \ln\left(\frac{K}{N_0}\right)$, donc :

$$\boxed{z : t \mapsto \ln\left(\frac{K}{N_0}\right) e^{-at}}.$$

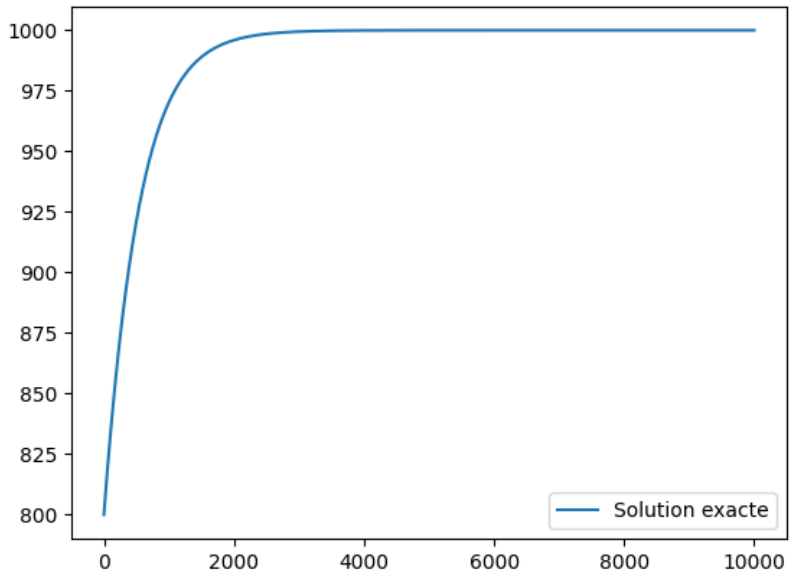
14. On en déduit :

$$\boxed{N : t \mapsto K\left(\frac{N_0}{K}\right)^{e^{-at}}}.$$

15. On en déduit que $\lim\limits_{t \to +\infty} N(t) = K$, ce qui justifie le fait que K est la capacité d'accueil du milieu.
16. On pose la fonction suivante :

```
def SolutionGompertz(t,N0):
    a=0.002
    K=1000
    return(K*(N0/K)**exp(-a*t))
```

On obtient le graphique suivant :

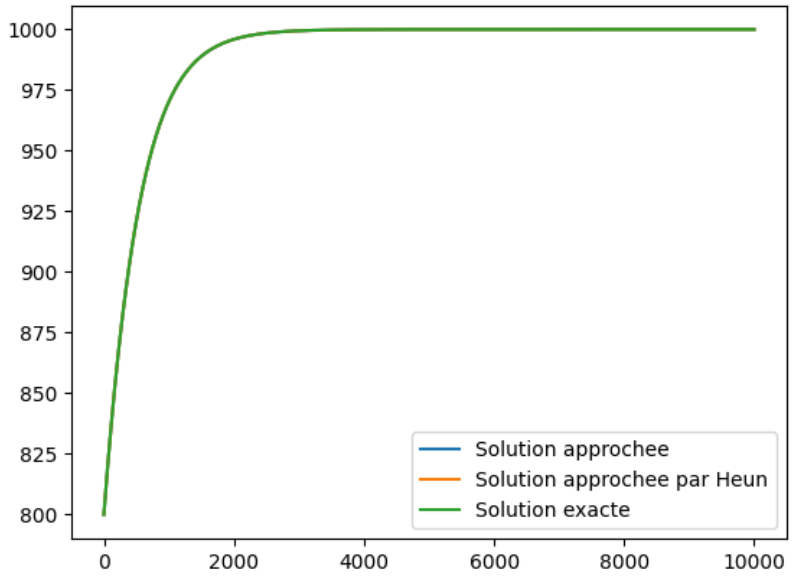


Là aussi, la population n'explose pas mais semble se stabiliser, le modèle est pertinent. L'évolution est beaucoup plus lente qu'avec le modèle logistique.

17. Il faut refaire les tests précédents en changeant la fonction :

```
def fGompertz(N):
    a=0.002
    K=1000
    return(a*N*log(K/N))
```

On obtient les graphiques suivants :



Là aussi, on a de bonnes approximations de la solution exacte, on ne distingue pas les courbes.

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 $\mathrm{dt} = \frac{t_f}{n}$ et on note pour tout $k \in [\![0, n]\!]$, $t_k = k\mathrm{dt}$.
La **méthode de Heun** consiste à poser :

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

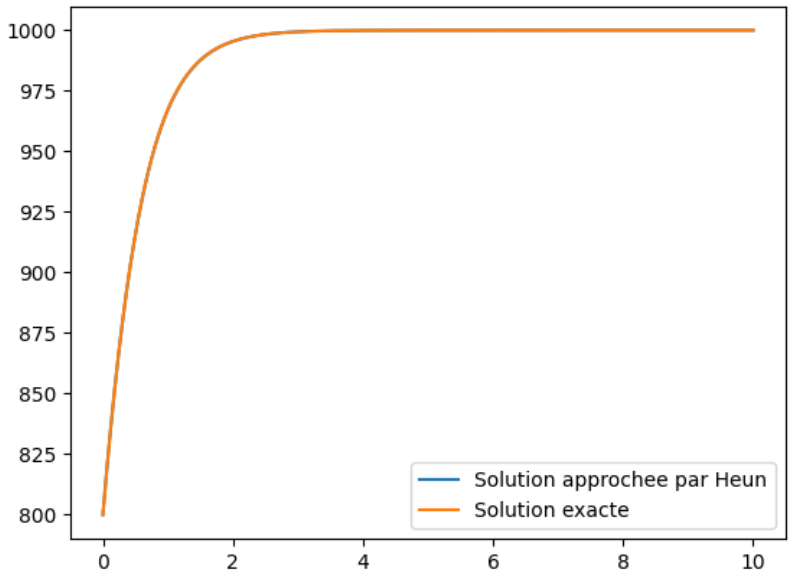
18. On écrit le programme suivant :

```
def Heun(fonc,tf,N0,dt):
    t=[0]
```

```
    N=[N0]
    while t[-1]+dt <= tf:
        k1=fonc(N[-1])*dt
        k2=fonc(N[-1]+k1)*dt
        x=N[-1]+(k1+k2)/2
        N.append(x)
        t.append(t[-1]+dt)
    return(t,N)
```

19. On teste et on trouve là aussi une très bonne approximation de la solution exacte :

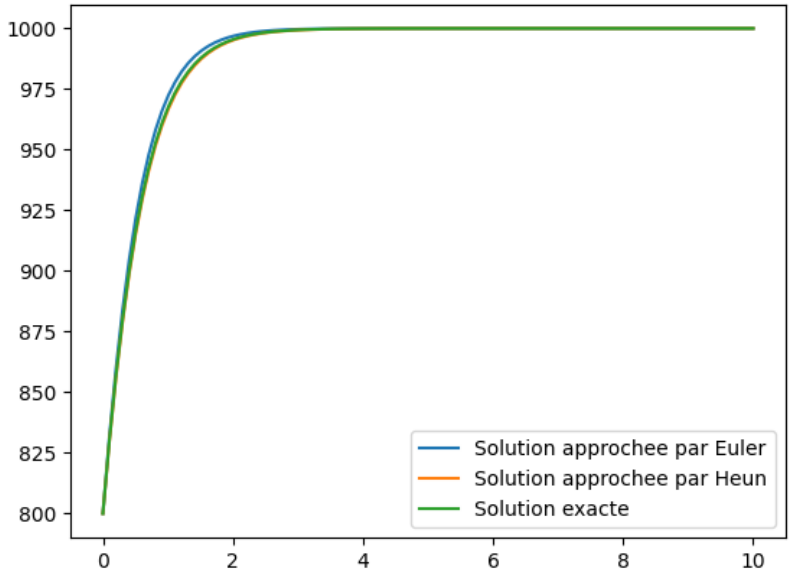
```
T,Approx2=Heun(fLogistique,tf,N0,dt)
plt.plot(T,Approx2,label="Solution approchee par Heun")

Vraie=[SolutionLogistique(t,N0) for t in T]
plt.plot(T,Vraie,label="Solution exacte")
plt.legend()
plt.show()
```



20. On trace le graphique :

```
tf=10
dt=0.1
N0=800

T,Approx=Euler(fLogistique,tf,N0,dt)
plt.plot(T,Approx,label="Solution approchee par Euler")

T,Approx2=Heun(fLogistique,tf,N0,dt)
plt.plot(T,Approx2,label="Solution approchee par Heun")

Vraie=[SolutionLogistique(t,N0) for t in T]
plt.plot(T,Vraie,label="Solution exacte")
plt.legend()
plt.show()
```

Pour dt = 0.1 on obtient alors les graphes suivants :



Il semble que la solution approchée par la méthode de Heun soit plus proche de la solution exacte.

21. On propose la fonction suivante :

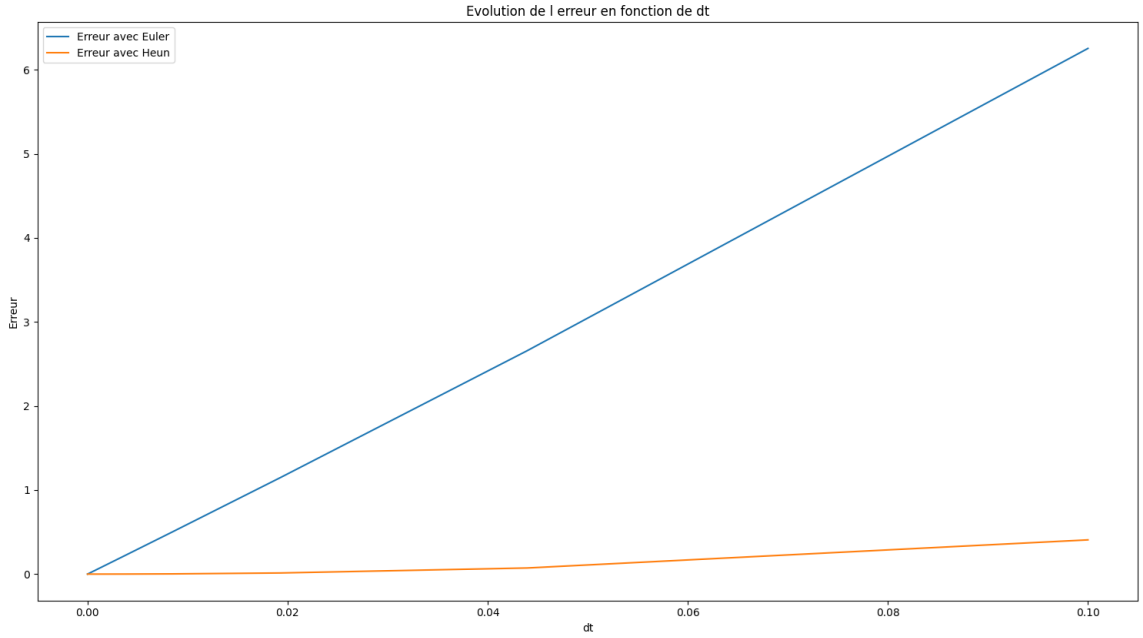
```
import nympy as np
def ErreurLogistique(tf,N0):
    dt=[10**(-k) for k in np.linspace(1,6,15)]
    X=[]
    for k in range(len(dt)):
        X.append(log(dt[k]))
    ErrEuler=[]
    ErrHeun=[]
    for k in dt:
        T,Approx=Euler(fLogistique,tf,N0,k)
        T,Approx2=Heun(fLogistique,tf,N0,k)
        Vraie=[SolutionLogistique(t,N0) for t in T]
        ErrEuler.append((max(abs(np.array(Approx)-np.array(Vraie)))))
        ErrHeun.append((max(abs(np.array(Approx2)-np.array(Vraie)))))
    return(dt,X,ErrEuler,ErrHeun)
```

`X` est là pour simplifier le tracé des courbes, il correspond au calcul de ln(dt).

22. On propose le code suivant :

```
tf=10
N0=800

dt,X,err1,err2=ErreurLogistique(tf,N0)
plt.plot(dt,err1,label="Erreur avec Euler")
plt.plot(dt,err2,label="Erreur avec Heun")
plt.legend()
plt.show()
```

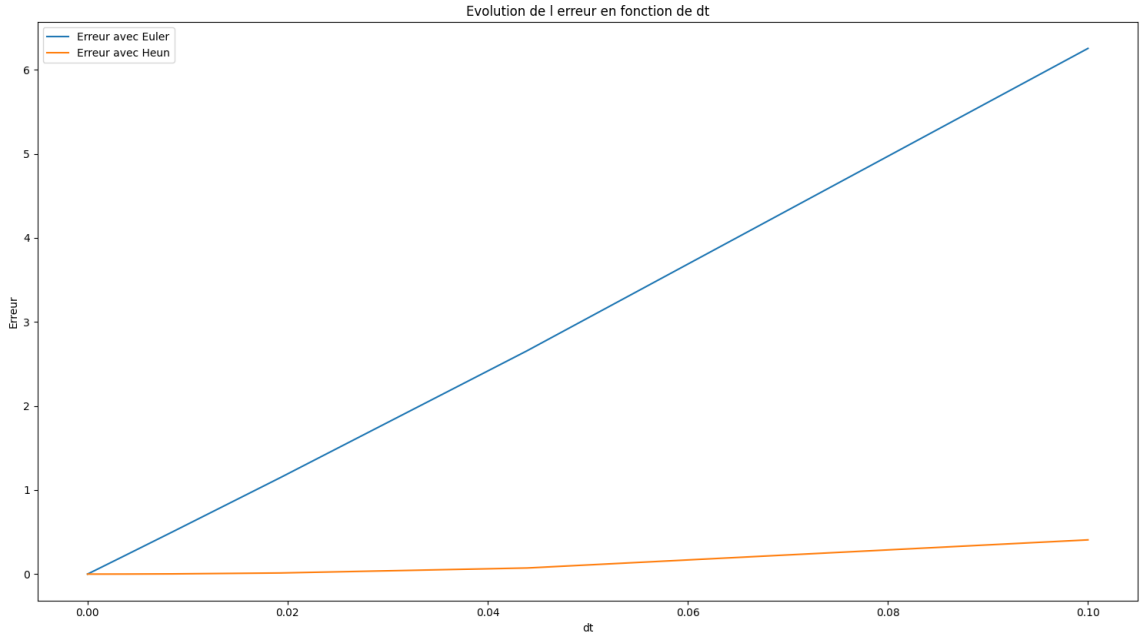
On obtient les courbes suivantes :



23. On s'en rend compte que l'échelle de l'axe des abscisses est complètement écrasée et qu'il serait plus pertinent de passer en échelle logarithmique :

```
tf=10
N0=800

dt,X,err1,err2=ErreurLogistique(tf,N0)
plt.plot(X,err1,label="Erreur avec Euler")
plt.plot(X,err2,label="Erreur avec Heun")
plt.legend()
plt.show()
```

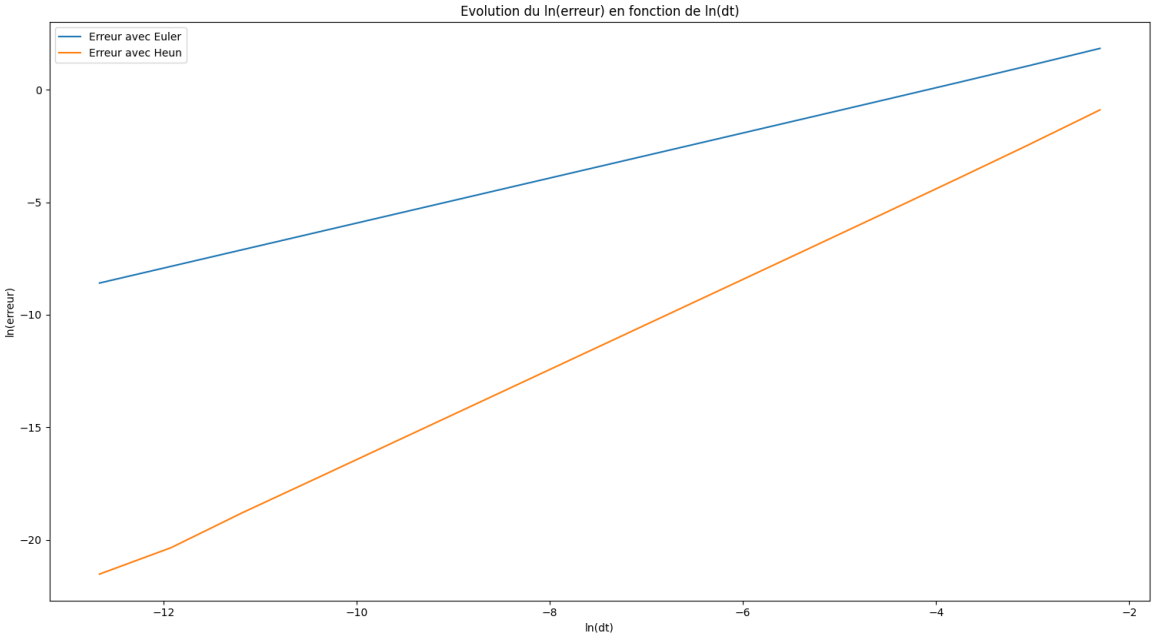
On obtient les courbes suivantes :



24. On modifie un peu nos fonctions :

```
def ErreurLogistiqueLog(tf,N0):
    dt=[10**(-k) for k in np.linspace(1,5.5,15)]
    X=[]
    for k in range(len(dt)):
        X.append(log(dt[k]))
    ErrEuler=[]
    ErrHeun=[]
    for k in dt:
        T,Approx=Euler(fLogistique,tf,N0,k)
        T,Approx2=Heun(fLogistique,tf,N0,k)
        Vraie=[SolutionLogistique(t,N0) for t in T]
        ErrEuler.append(log(max(abs(np.array(Approx)-np.array(Vraie)))))
        ErrHeun.append(log(max(abs(np.array(Approx2)-np.array(Vraie)))))
    return(X,ErrEuler,ErrHeun)

X,err1,err2=ErreurLogistiqueLog(tf,N0)
plt.plot(X,err1,label="Erreur avec Euler")
plt.plot(X,err2,label="Erreur avec Heun")
plt.xlabel('ln(dt)')
plt.ylabel('ln(erreur)')
plt.title('Evolution du ln(erreur) en fonction de ln(dt)')
plt.legend()
plt.show()
```

On obtient les courbes suivantes :



On a deux droites. On détermine leur pente par le code suivant :

```
# Regression linéaire pour les pentes des droites
a1=scipy.stats.linregress(X,err1)
a2=scipy.stats.linregress(X,err2)
print(a1[0],a2[0])
```

On obtient :

```
>>> (executing lines 81 to 83 of "Eul
er.py")
1.0034021682914172 2.0013182285078583
```

Si on arrondit, on trouve donc une pente de 1 pour la méthode d'Euler et de 2 pour la méthode de Heun.
Ainsi, on a montré numériquement pour la méthode d'Euler :

$$\ln(\text{erreur}) = 1 \times \ln(\text{dt}) + b \iff \text{erreur} = e^b \times \text{dt}$$

et pour la méthode de Heun :

$$\ln(\text{erreur}) = 2 \times \ln(\text{dt}) + c \iff \text{erreur} = e^c \times \text{dt}^2.$$

On dit que la méthode d'Euler est d'ordre 1 et que la méthode de Heun est d'ordre 2.