éduSCOL
Informer et accompagner les professionnels de l'éducation

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE DE L'ÉDUCATION NATIONALE ET DE LA JEUNESSE

VOIE GÉNÉRALE

2DE | 1RE | TLE

Physique-chimie

ENSEIGNEMENT SPÉCIALITÉ

PROGRAMMER EN PYTHON
FICHE POUR ALLER PLUS LOIN N°1 : MÉTHODE D'EULER ET THERMALISATION

L'étude de la thermalisation d'un système incompressible en contact avec un thermostat à température T_s permet de mettre en place une résolution d'équation différentielle basée sur la méthode d'Euler.

> **Capacité numérique mise en œuvre**
> Utiliser un langage de programmation pour résoudre une équation différentielle grâce à la méthode d'Euler.

L'équation différentielle est ici vue par l'intermédiaire d'une fonction f(T,t). L'idée générale est de l'écrire dT/dt=f(T,t). On commence donc par définir cette fonction.

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

```
t0, tf = 0., 10. # bornes de l'intervalle de temps pour la résolution
N = 1000 # nombre de pas
t = np.linspace(t0,tf,N+1) # liste des dates pour le calcul des solutions aprochées
T0=300 #intialisation x(0)=0
a=1
c=1
TS=400
def f(T,t):
    return -a*(T-TS)/c
```

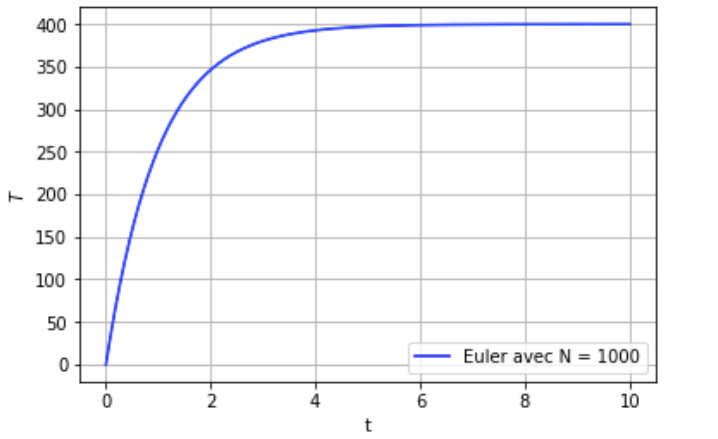
On met ensuite en place une fonction Euler qui permet, comme dans la méthode classique, de calculer T(i+1) connaissant T(i), à l'aide de la fonction f.

Il faut pour cela créer une liste de valeur de T, que l'on initie à l'aide de la valeur initiale.

```
def euler(g,x0,t): # méthode d'Euler
    h = t[1]-t[0] # pas constant, choisi comme distance entre les deux premières valeurs de
    N = len(t)  # on recalcule N en local pour le cas général
    x = np.zeros(N) # intialisation obligatoire du tableau numpy, crée un tableau de longue
    x[0] = x0 # initialisation de la résolution
    for i in range(N-1):
        x[i+1] = x[i]+h*g(x[i],t[i]) # définit x[i+1] par Euler
    return x # la boucle précédente sert à calculer les valeurs de x, la fin définit x comm
```

On peut alors faire tracer l'évolution de la température en fonction du temps.

```
solE = euler(f,x0,t)
plt.figure()
plt.plot(t,solE,'b-',label='Euler avec N = '+str(N))
plt.legend(loc='best')
plt.xlabel('t') ; plt.ylabel('$T$')
plt.grid()
plt.show()
```



Il est ensuite intéressant de faire constater que le découpage temporel a une importance cruciale en simulation numérique. On peut ici faire recalculer l'évolution de la température avec seulement 15 pas.

```
t20, t2f = 0., 10. # bornes de l'intervalle de temps pour la résolution
M = 15 # nombre de pas
t2 = np.linspace(t20,t2f,M+1) # liste des dates pour le calcul des solutions aprochées


def euler2(g,x0,t2): # méthode d'Euler
    h = t2[1]-t2[0] # pas constant, choisi comme distance entre les deux premières valeurs
    M = len(t2)  # on recalcule N en local pour le cas général
    x = np.zeros(M) # intialisation obligatoire du tableau numpy, crée un tableau de longue
    x[0] = x0 # initialisation de la résolution
    for i in range(M-1):
        x[i+1] = x[i]+h*g(x[i],t2[i]) # définit x[i+1] par Euler
    return x # la boucle précédente sert à calculer les valeurs de x, la fin définit x comm

solE = euler(f,T0,t)
solE2=euler2(f,T0,t2)
plt.figure()
plt.plot(t,solE,'b-',label='Euler avec N = '+str(N))
plt.plot(t2,solE2,'ro',label='Euler avec N = '+str(M))
plt.legend(loc='best')
plt.xlabel('t') ; plt.ylabel('$T$')
plt.grid()
plt.show()
```

Le tracé permet de comparer la qualité des deux simulations. On voit qu'avec seulement 15 points, la simulation est significativement différente de celle qui met en jeu 1000 points.