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°3 : MÉTHODE D'EULER ET CHARGE D'UN CONDENSATEUR

L'étude de la charge d'un condensateur permet de mettre en place une résolution d'équation différentielle fondée sur la méthode d'Euler et de comparer les résultats d'une intégration numérique à ceux d'une méthode analytique afin de juger de la qualité de la méthode d'Euler. Les méthodes utilisées dans cette fiche sont très proches de celles mises en place dans la fiche n°1.

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

L'équation différentielle vérifiée par la tension U(t) aux bornes du condensateur est ici vue par l'intermédiaire d'une fonction f(U,t). L'idée générale est de l'écrire dU/dt = f(U,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
x0=0 #intialisation x(0)=0
E=10
Tau=1
def f(U,t):
    return (E-U)/Tau
```

On met ensuite en place une fonction Euler qui permet, comme dans la méthode classique, de calculer U(i+1) connaissant U(i), à l'aide de la fonction f.

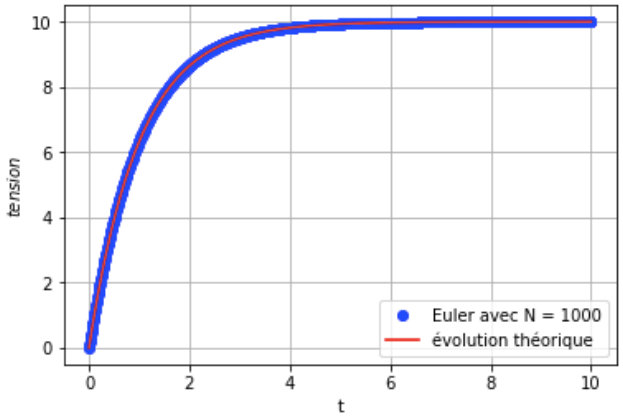
Il faut pour cela créer une liste de valeur de U, que l'on initie à l'aide de la valeur initiale.

```
def euler(g,x0,t): # méthode d'Euler
    h = t[1]-t[0] # pas constant, choisi comme distance entre les deux premières valeurs de la liste de t
    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 longueur N rempli de 0
    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 comme fonction solution
```

On peut alors faire tracer l'évolution de la tension U(t) en fonction du temps. Sur le même graphe, on peut faire tracer l'évolution théorique de U(t), en exponentielle relaxante. Cela permet de constater que pour N = 1000 points de calcul, la méthode d'Euler semble correcte.

```
def Uthéo(t):
    return E*(1-np.exp(-t/Tau))

solE = euler(f,x0,t)
plt.figure()
plt.plot(t,solE,'bo',label='Euler avec N = '+str(N))
plt.plot(t,Uthéo(t),'r-',label='évolution théorique')
plt.legend(loc='best')
plt.xlabel('t') ; plt.ylabel('$tension$')
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 tension avec seulement 15 pas. On voit qu'avec seulement 15 points, la simulation est significativement différente de l'évolution théorique. C'est une des limites la méthode d'Euler.