Capacité numérique : Mettre en œuvre la méthode d'Euler à l'aide d'un langage de programmation pour simuler la réponse d'un système linéaire du premier ordre à une excitation de forme quelconque

On a vu que lors de la charge d'un condensateur C par une tension E la tension u(t) aux bornes du condensateur vérifiait l'équation différentielle :

$$\dot{u} + \frac{u}{\tau} = \frac{E}{\tau}, \text{ soit encore : } \dot{u} = \frac{E-u}{\tau}$$

La méthode d'Euler est une méthode de résolution approchée, en proposant une expression approchée de la dérivée :

$$\dot{u} = \frac{du}{dt} \approx \frac{\Delta u}{\Delta t}$$

On pose un intervalle de temps petit – le pas Δt. Alors sur cet intervalle de temps Δt, la tension u a varié de Δu = u(t+Δt) – u(t)

Ici : $\dot{u} = f(u)$ où $f(u) = \frac{E-u}{\tau}$

$$\frac{u(t+\Delta t) - u(t)}{\Delta t} = f(u)$$

$$u(t+\Delta t) = u(t) + f(u).\Delta t$$

Ainsi on va découper l'intervalle de temps $[t_0 ; t_f]$ en n intervalles de même longueur Δt. On dispose ainsi de n+1 valeurs du temps $t_k = t_0 + k\,\Delta t$ pour $k \in \{0,...,n\}$. On va alors approximer la solution u à l'instant t_k par le nombre u_k défini par la relation de récurrence : $u_{k+1} = u_k + f(u_k).\Delta t$

On peut alors faire tracer l'évolution de la tension u(t) en fonction du temps, point par point.

Dans le script Python suivant, on a pris t0 = 0, E = 10 V, tau = 0,01 s et tf = 0,05 s. On a défini une fonction d'Euler qui dépendant du nombre n choisi.

```
# Méthode d'Euler

# On importe les bibliothèques
import numpy as np
import matplotlib.pyplot as plt

# On introduit les valeurs numériques
t0 = 0
E = 10
tau = 0.01
u0 = 0
tf = 0.05


# On définit la fonction F
def F(u):
 return (E - u) / tau

# On définit la fonction d'Euler
def Euler( n):
    pas = (tf - t0) / n
    u = u0
    U = [u0]
    t = t0
    T = [t0]
    for k in range(n):      # n intérations, donc n+1 points
            u = u + pas * F(u)
            t = t + pas
            U.append(u)
            T.append(t)
    return U, T

U, T = Euler(10)

# On trace la courbe obtenue
plt.plot(T, U,'r+-')
plt.xlabel("temps t")
plt.ylabel("tension u(t)")
plt.show()
```

Sur le même graphe, on peut faire tracer l'évolution théorique de u(t), puisque le calcul a été fait en cours :

$$u(t) = E\left(1 - e^{-t/\tau}\right)$$

```
# On compare avec la courbe attendue
temps = []
Utheo = []
for t in np.linspace(0, tf,101):
    utheo = E* (1 - np.exp(t/(-tau)))
    temps.append(t)
    Utheo.append(utheo)
plt.plot(temps, Utheo, 'b')
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 différents pas (10 ou 100 par exemple). On voit qu'avec seulement 10 points, la simulation est significativement différente de l'évolution théorique. C'est une des limites la méthode d'Euler.