8 Équation différentielle d'ordre 1

Capacités exigibles

- Mettre en œuvre la méthode d'Euler explicite afin de résoudre une équation différentielle d'ordre 1
- Utiliser la fonction **odeint** de la bibliothèque **scipy.integrate** (sa spécification étant fournie)

A Méthode d'Euler explicite

A.1 Principe

Soit $y(t)$ une fonction inconnue et vérifiant une équation du 1er ordre pouvant s'écrire sous la forme :

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(y,t)$$

où $f(y,t)$ est une expression faisant intervenir directement y et éventuellement la variable de temps t

Exemple

La tension $u(t)$ aux bornes du condensateur du circuit RC série soumis à un échelon de tension E vérifie :

$$\tau\dot{u} + u = E$$

Cette équation peut s'écrire sous la forme :

$$\frac{\mathrm{d}u}{\mathrm{d}t} = f(u,t) \qquad \text{où} \quad f(u,t) = \frac{E-u}{\tau}$$

On cherche à tracer la fonction $y(t)$ entre deux instants t_i et t_f, connaissant la condition initiale $y(t_i) = y_i$.
La méthode d'Euler explicite consiste dans un premier temps à **subdiviser l'axe temporel** en N petits intervalles de temps de longueur $h = \dfrac{t_f - t_i}{N}$ (le pas de temps). On dit que l'on discrétise l'axe des temps. Notons $t_k = t_i + kh$ les instants successifs étudiés. Ainsi, $t_i = t_0$ et $t_f = t_N$.

Puis, **on intègre l'équation différentielle** $\dfrac{\mathrm{d}y}{\mathrm{d}t} = f(y,t)$ sur l'intervalle $[t_k, t_{k+1}]$:

$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} f(y,t)\,\mathrm{d}t$$

pour faire apparaître une **relation de récurrence**, en notant $y_k = y(t_k)$:

$$y_{k+1} = y_k + \int_{t_k}^{t_{k+1}} f(y,t)\,\mathrm{d}t$$

Ainsi, connaissant $y_0 = y_i$, **on peut en déduire pas à pas toutes les valeurs y_k** grâce à la relation ci-dessus.

Mais encore faut-il pouvoir préciser la valeur de l'intégrale $\displaystyle\int_{t_k}^{t_{k+1}} f(y,t)\,\mathrm{d}t$. Pour cela, la méthode d'Euler explicite consiste à considérer que la fonction $f(y,t)$ prend la valeur constante $f(y_k,t_k)$ sur l'intervalle $[t_k, t_{k+1}]$:

$$\forall t \in [t_k, t_{k+1}]\,,\ f(y,t) = f(y_k,t_k)$$

R *Approximation similaire à la méthode des rectangles à gauche, voir partie 7.*

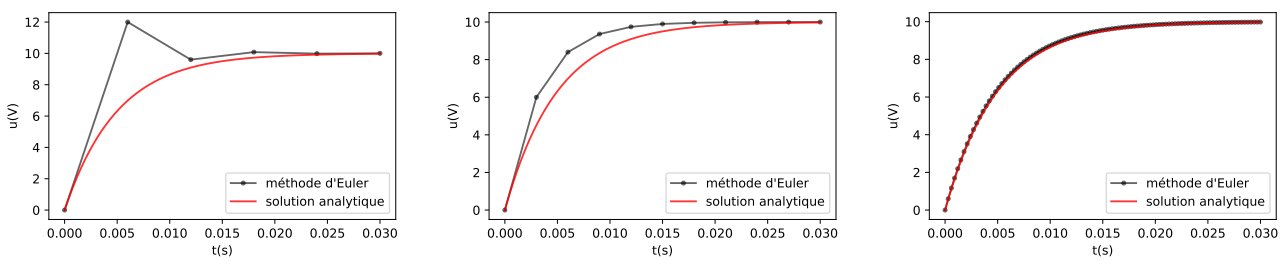
D'où $\displaystyle\int_{t_k}^{t_{k+1}} f(y,t)\,\mathrm{d}t \simeq f(y_k,t_k)\,h$. On obtient alors la relation de récurrence suivante :

$$\boxed{y_{k+1} = y_k + f(y_k,t_k)\,h}$$

A.2 Implémentation sous Python et exemple

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

# --------------------------------
# ALGORITHME D'EULER

def euler(f,yi,ti,tf,N):
    # pas de temps
    h = (tf-ti)/N

    # conditions initiales :
    t = ti
    y = yi

    # on initialise les listes qui accueilleront les valeurs successives
    # de t et y obtenues grâce à la boucle for
    liste_t = [t]
    liste_y = [y]

    for k in range(1,N+1):
        # application des relations de récurrence
        y = y + f(y,t)*h
        t = t + h

        # on remplit les listes à chaque itération
        liste_t.append(t)
        liste_y.append(y)

    return liste_t, liste_y

# --------------------------------
# CIRCUIT RC SERIE SOUMIS A UN ECHELON DE TENSION

E = 10 #V
R = 500 #Ohms
C = 10E-6 # F
tau = R*C

def f_RC_echelon(u,t):
    return (E-u)/tau

ti = 0
ui  = 0 # condensateur initialement déchargé
tf = 6*tau # supérieur à 5*tau, donc régime transitoire quasi-terminé
N = 100

t_euler,u_euler = euler(f_RC_echelon,ui,ti,tf,N)

plt.plot(t_euler,u_euler, marker='.',label = "méthode d'Euler")

# solution analytique pour comparer
ta = np.linspace(ti,tf,1000)
ua = E*(1-np.exp(-ta/tau))
plt.plot(ta,ua,label="solution analytique")

plt.legend(loc='lower right')
plt.xlabel('t(s)')
plt.ylabel('u(V)')
plt.show()
```

$N = 5$ $N = 10$ $N = 100$



La méthode d'Euler est d'autant plus satisfaisante que le pas de temps est faible (càd N élevé).

B Utilisation de la fonction odeint

La résolution d'une équation différentielle d'ordre 1 de la forme :

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(y,t)$$

peut aussi s'effectuer avec la fonction **odeint** de la bibliothèque **scipy.integrate**.

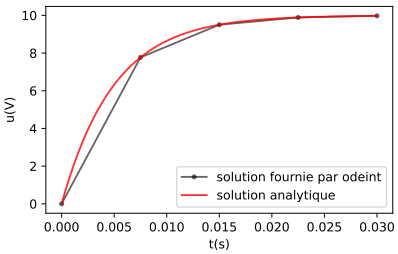
Il faut préalablement définir une échelle de temps, par exemple avec **`np.linspace`** et utiliser l'instruction suivante (en ayant préalablement chargé la bibliothèque scipy.integrate avec l'alias sp par exemple) :

`sp.odeint(`*fonction f, valeur initiale de $y(t)$, array des N instants t_k***`)`**

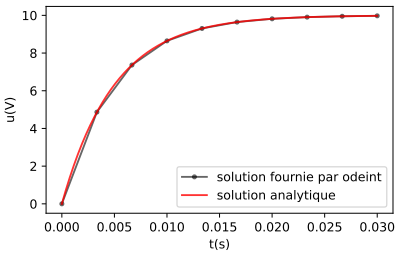
Reprenons l'exemple précédent pour illustrer. On rajoute les lignes suivantes pour cela :

```
60 import scipy.integrate as sp
61
62 # solution avec odeint
63
64 t_odeint = np.linspace(ti,tf,N)
65 u_odeint = sp.odeint(f_RC_echelon,ui,t_odeint)
66
67 plt.plot(t_odeint,u_odeint,marker='.',label="solution fournie par odeint")
68 plt.legend(loc='lower right')
69 plt.show()
```

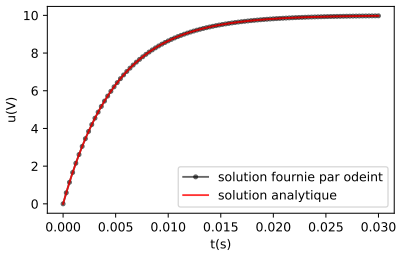
$N = 5$



$N = 10$



$N = 100$



La solution fournie par odeint est très satisfaisante même si le pas de temps est grand (càd N faible).

R *L'algorithme mis en œuvre par odeint est bien plus élaboré et robuste que le simple algorithme d'Euler, mais conformément au programme, nous ne rentrerons pas dans les détails...*

Bien entendu, il vaut tout de même mieux choisir un grand nombre N de points afin de lisser le tracé.