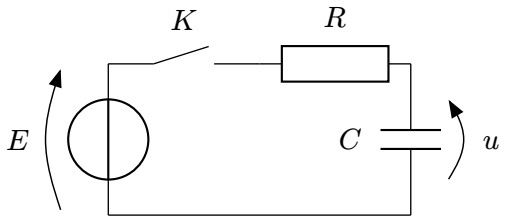
Méthodes numériques
Résolution numérique d'équations différentielles
TD

1. 💻 Circuit RC

On cherche à modéliser la charge d'un condensateur dans un circuit RC série soumis à une tension continue E.



À $t = 0$, l'interrupteur K est fermé. Le condensateur est initialement déchargé.

1/ Montrer que u vérifie l'équation différentielle

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{1}{RC}u + \frac{1}{RC}E$$

La loi des mailles s'écrit $E = u_R + u$ avec $u_R = Ri$ et $i = C\frac{\mathrm{d}u}{\mathrm{d}t}$. Donc

$$E = RC\frac{\mathrm{d}u}{\mathrm{d}t} + u$$

2/ Que vaut $u(t = 0^+)$?

La tension aux bornes du condensateur est continue, de plus elle est nulle à $t = 0^-$ (condensateur initialement déchargé). Donc $u(0^+) = 0$.

On souhaite résoudre numériquement cette équation différentielle à l'aide de la méthode d'Euler explicite. On note $u_i = u(i \cdot \Delta t)$ la tension discrétisée.

3/ Établir une relation de récurrence sur u_i.

La relation de Taylor à l'ordre 1 donne

$$u_{i+1} = u(i \cdot \Delta t + \Delta t) = u(i \cdot \Delta t) + \Delta t\frac{\mathrm{d}u}{\mathrm{d}t} = u_i - \Delta t\left(-\frac{1}{RC}u_i + \frac{1}{RC}E\right)$$

4/ Écrire une suite d'instructions permettant de calculer les valeurs successives de u_i. On prendra $R = 1\,\mathrm{k\Omega}$, $C = 1\,\mu\mathrm{F}$, $E = 5\,\mathrm{V}$, $\Delta t = 0.2\,\mathrm{ms}$ et on effectuera 25 itérations.

```
R = 1e3
C = 1e-6
E = 5
Delta_t = 0.2e-3

N = 25
u = [0]  # u(0) = 0 V
```

```
for i in range(N):
    u.append(u[i] + Delta_t * (-1/(R * C) * u[i] + 1/(R * C) * E))
```

5/ Tracer l'évolution de u en fonction du temps. Comparer avec la solution analytique que vous calculerez et tracerez également.

La solution analytique est

$$u(t) = E\left(1 - \exp\left(-\frac{t}{RC}\right)\right)$$

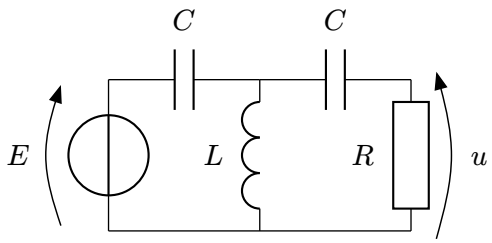
```
import numpy as np
import matplotlib.pyplot as plt

temps = [i * Delta_t for i in range(N + 1)]
u_analytique = [E * (1 - np.exp(-t / (R * C))) for t in temps]

plt.plot(temps, u, label='Numérique (Euler explicite)')
plt.plot(temps, u_analytique, label='Analytique')
plt.xlabel('Temps (s)')
plt.ylabel('Tension u (V)')
plt.legend()
plt.show()
```

6/ Observer qualitativement l'effet de la valeur de Δt sur la précision de la solution numérique.

Plus Δt est petit, plus la solution numérique est précise et proche de la solution analytique.

2. 💻 Circuit électrique d'ordre 3

On s'intéresse au circuit électrique d'ordre 3 représenté ci-dessous.



Les valeurs des composants sont $R = 10\,\Omega$, $L = 1 \cdot 10^{-3}\,\mathrm{H}$ et $C = 1 \cdot 10^{-6}\,\mathrm{F}$. La source de tension fournit une tension constante $E = 10\,\mathrm{V}$.

L'évolution de u est régie par l'équation différentielle

$$RLC^2\frac{\mathrm{d}^3u}{\mathrm{d}t^3} + 2LC\frac{\mathrm{d}^2u}{\mathrm{d}t^2} + RC\frac{\mathrm{d}u}{\mathrm{d}t} + u(t) = 0$$

On suppose que les conditions initiales sont $u(t=0) = 0$, $\frac{\mathrm{d}u}{\mathrm{d}t}\Big|_{t=0} = 3\,\mathrm{V\,s^{-1}}$ et $\frac{\mathrm{d}^2u}{\mathrm{d}t^2}\Big|_{t=0} = 0$. L'objectif de cet exercice est de déterminer l'évolution de $u(t)$ au cours du temps.

1/ On note $v(t) = \frac{\mathrm{d}u}{\mathrm{d}t}$ et $w(t) = \frac{\mathrm{d}^2u}{\mathrm{d}t^2}$. Mettre le problème sous la forme d'un problème d'Euler en exprimant $\frac{\mathrm{d}u}{\mathrm{d}t}$, $\frac{\mathrm{d}v}{\mathrm{d}t}$ et $\frac{\mathrm{d}w}{\mathrm{d}t}$ en fonction de $u(t)$, $v(t)$ et $w(t)$.

$$\begin{cases} \frac{\mathrm{d}u}{\mathrm{d}t} = v \\ \frac{\mathrm{d}v}{\mathrm{d}t} = \frac{\mathrm{d}^2u}{\mathrm{d}t^2} = w \\ \frac{\mathrm{d}w}{\mathrm{d}t} = \frac{\mathrm{d}^3u}{\mathrm{d}t^3} = \frac{-2LCw(t) - RCv(t) - u(t)}{RLC^2} \end{cases}$$

2/ On note `Y np.array([u, v, w])`. Définir une fonction `dY_dt(Y,t)` qui retourne le tableau $\frac{\mathrm{d}Y}{\mathrm{d}t}$ en prenant comme entrée le temps t et le tableau Y.

```
def dY_dt(Y, t):
    u, v, w = Y
    du_dt = v
    dv_dt = w
    dw_dt = (-2 * L * C * w - R * C * v - u) / (R * L * C**2)
    return np.array([du_dt, dv_dt, dw_dt])
```

3/ Écrire une suite d'instructions permettant de calculer l'évolution de $Y(t)$ entre $t = 0$ et $t = 1\,\mathrm{ms}$ avec un pas de temps $\Delta t = 1\,\mu\mathrm{s}$ en utilisant la méthode d'Euler explicite.

```
# Définir les constantes
R = 10  # ohm
L = 1e-3  # H
C = 1e-6  # F
E = 10  # V

# Conditions initiales
u0 = 0  # V
v0 = 3  # V/s
w0 = 0  # V/s²
Y0 = np.array([u0, v0, w0])

# Paramètres de temps
t_final = 1e-3  # s
dt = 1e-6  # s

# Listes pour stocker les résultats
temps = [0]
Y = [Y0]

while temps[-1] < t_final:
    Y.append(Y[-1] + dY_dt(Y[-1], temps[-1]) * dt)
    temps.append(temps[-1] + dt)
```

4/ Tracer l'évolution de $u(t)$ au cours du temps.

```
import matplotlib.pyplot as plt

u = [y[0] for y in Y]

plt.plot(temps, u)
plt.xlabel('Temps (s)')
plt.ylabel('Tension u(t) (V)')
plt.title('Évolution de la tension u(t) au cours du temps')
plt.grid()
plt.show()
```

La fonction `odeint(func, y0, t)`[1] de la bibliothèque `scipy.integrate` permet de résoudre des problèmes d'Euler en appliquant des méthodes plus sophistiquées mais reposant sur le même principe. Elle prend en entrée

[1]Documentation complète disponible à l'adresse https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html

- `func`: une fonction qui retourne la dérivée de l'état en fonction de l'état et du temps,
- `y0`: l'état initial,
- `t`: un tableau des instants où l'on souhaite connaître la solution.

5/ Reprendre la question 3 en utilisant la fonction `odeint` pour calculer l'évolution de $Y(t)$.

```
from scipy.integrate import odeint

# Paramètres de temps
t_final = 1e-3  # s
dt = 1e-6  # s
t = np.arange(0, t_final, dt)

# Calcul de l'évolution de Y(t) avec odeint
Y = odeint(dY_dt, Y0, t)
```

3. 💻 Tir cadré ? ⋆

On étudie un tir au football. La vitesse initiale du ballon est de $20\,\mathrm{m\,s^{-1}}$ selon l'axe x (horizontal) et de $12\,\mathrm{m\,s^{-1}}$ selon l'axe z (vertical). Le ballon est sur le sol juste avant le tir.

Dans un premier temps, on ne prend en compte que la gravité.

1/ Établir l'équation différentielle vérifiée par la vitesse $\vec{v}$ du ballon. Exprimer la dérivée de la vitesse.

Le théorème de la résultante cinétique appliqué au ballo, s'écrit

$$m\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = m\vec{g}$$

soit

$$\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = \vec{g}$$

On résout numériquement l'équation différentielle en utilisant la fonction `solve_ivp` de la bibliothèque `scipy.integrate`.

2/ Compléter le code suivant.

```
from scipy.integrate import solve_ivp, trapezoid
import numpy as np

g = 9.81 # m/s^2
rho = 1.2 # kg/m^3
v0 = np.array([20, 0, 12]) # m/s

def dv_dt(t, v):
    a = ... # accélération
    return a

sol = solve_ivp(dv_dt, [0, 2], v0, max_step=0.01)

t = sol.t
vx = sol.y[0, :]
vy = sol.y[1, :]
vz = sol.y[2, :]
```

```
def dv_dt(t, v):
    a = - g * np.array([0,0,1])
    return a
```

Il est maintenant nécessaire de calculer la position du ballon en intégrant la vitesse en utilisant la méthode des rectangles.

3/ Compléter le code suivant. Attention, les temps calculés par la fonction `solve_ivp` ne sont pas forcément régulièrement espacés.

```
x = [0]
y = [0]
z = [0]

for i in range(1, len(sol.t)):
    x.append(...)
    y.append(...)
    z.append(...)
```

```
for i in range(1, len(sol.t)):
    x.append(x[-1] + vx[i] * (t[i]-t[i-1]))
    y.append(y[-1] + vy[i] * (t[i]-t[i-1]))
    z.append(z[-1] + vz[i] * (t[i]-t[i-1]))
```

Pour vérifier si le tir est cadré, on trace la trajectoire du ballon grâce au code suivant.

```
fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")

ax.plot(x, y, z)

# Tracé des cages
x_cages = 14.5
y_cages = 0.3
l_cages = 7.32
h_cages = 2.44
ax.plot(
    [x_cages]*4,
    [y_cages, y_cages, y_cages + l_cages, y_cages + l_cages],
    [0, h_cages, h_cages, 0],
    color="orange")

# Mise en forme
ax.set_xlabel("x (m)")
ax.set_ylabel("y (m)")
ax.set_zlabel("z (m)")
ax.set_title("Trajectoire 3D")
ax.set_zlim(0, max(z)*1.1)

plt.tight_layout()
plt.show()
```

4/ Le tir est-il cadré ?

Non, la trajectoire du ballon passe au-dessus des cages.

On prend maintenant en compte les frottements avec l'air $\vec{F} = -\frac{1}{2}\pi\rho C_x R^2 \|\vec{v}\| \vec{v}$. On donne les valeurs numériques suivantes : $\rho = 1.2\,\mathrm{kg\,m^{-3}}$, $C_x = 0.47$, $R = 0.11\,\mathrm{m}$ et $m = 145\,\mathrm{g}$

5/ Modifier la fonction `dv_dt` pour inclure la force de traînée.

On pourra utiliser la fonction `np.linalg.norm(v)` pour calculer la norme du vecteur vitesse v.

Le tir est-il maintenant cadré ?

```
rho = 1.2 # kg/m^3
Cx = 0.47 # coefficient de trainée
```

```
R = 0.11 # rayon m
S = np.pi * R**2 # cross-sectional area
m = 0.145 # masse kg

def dv_dt(t, v):
    a = - g * np.array([0,0,1]) - 1/2 * rho * Cx * S / m * np.linalg.norm(v) * v
    return a
```

Le tir n'est toujours pas cadré, il passe à coté des cages.

Le footballeur a mis de l'effet dans la balle en lui imprimant une rotation de $\Omega = 100$ min^{-1} autour de l'axe (Oz). Cette rotation engendre une force de portance appelée force de Magnus et s'exprimant comme $\frac{1}{2}C\rho R^3\vec{\Omega} \wedge \vec{v}$ avec $C \approx 1$.

6/ Modifier la fonction `dv_dt` pour inclure la force de Magnus.

Le tir est-il maintenant cadré ?

```
Omega = np.array([0,0,100*2*np.pi/60])

def dv_dt(t, v):
    a = - g * np.array([0,0,1]) - 1/2 * rho * Cx * S / m * np.linalg.norm(v) * v + 1/2 * rho * R**3 *
np.cross(Omega,v) / m
    return a
```

Oui, en prenant en compte toutes ces forces, le tir est cadré.