Corrigé du Devoir à la maison 5

Réponse indicielle du circuit *RC*

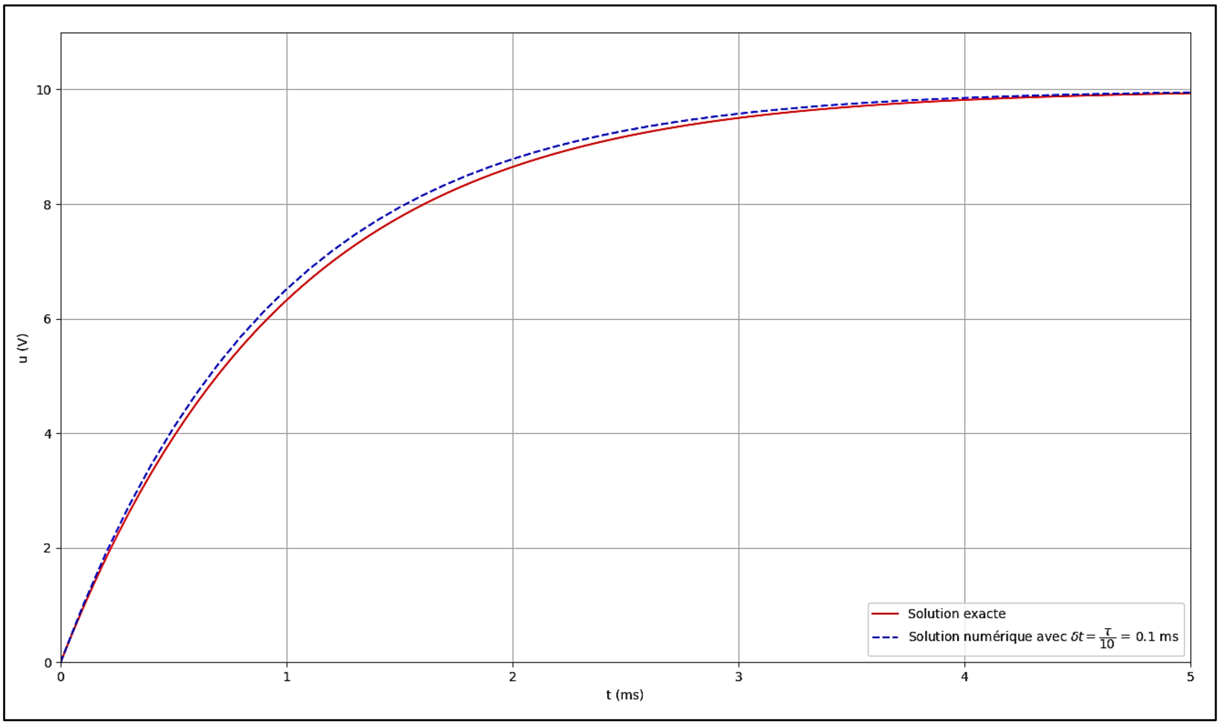
```python
## Cellule 1 : Importation des bibliothèques nécessaires

import numpy as np                  # pour la manipulation des tableaux
import matplotlib.pyplot as plt     # pour les représentations graphiques

## Cellule 2 : Système physique étudié et valeurs numériques des paramètres physiques
# Valeurs numériques de paramètres physiques
E = 10                    # en V
R = 1.e4                  # en Ohm
C = 100.e-9               # en F
tau = R*C                 # Expression littérale (en s)
t0, tf = 0 , 5* tau       # bornes de l'intervalle de résolution (en s)
u0 = 0                    # Condition initiale (en V)

# Équation différentielle à résoudre
def derivee_u(u, t):
    """
    Fonction explicitant du/dt en fonction de u et t .
    """
    return E/tau - u/tau

```

```python
## Cellule 3 : Implémentation de la méthode d'Euler explicite
def euler(F, y0, t0, tf, dt):
    """
    Résout le problème de Cauchy y'(t)=F(y(t),t) avec y(0)=y0 par la méthode
    d'Euler explicite.
    Arguments d'entrée :
    - F : fonction donnant y' (fonction de 2 variables) ;
    - y0 : condition initiale sur y (flottant) ;
    - t0 et tf : bornes de l'intervalle de résolution (flottants) ;
    - dt : pas de discrétisation utilisé pour la résolution (flottant).
    Variables de sortie :
    - t : vecteur contenant l'ensemble des instants tk (array numpy) ;
    - y : vecteur contenant l'ensemble des valeurs approchées yk (array numpy).
    """
  # Création et initialisation des variables de sortie
    """
    t = np.arange(tmin,tmax,dt)
    Renvoie un tableau de points, espacés du pas dt, entre tmin (inclus) et tmax (exclus)
    """
    t = np.arange(t0, tf+dt, dt)   # les valeurs sont comprises dans l'intervalle [t0,tf+dt[ avec un pas égal à dt
    N = len(t)                     # Taille du tableau t
    y = np.zeros(N)                # initialisation du tableau y
    y[0] = y0                      # prise en compte de la CI

  # Boucle permettant le calcul des yk par récurrence
    for k in range(0,N-1):
        y[k+1] = y[k] + F(y[k],t[k])*dt

    return t, y

```

```python
## Cellule 4 : Résolution numérique
dt = tau / 10                          # Expression du pas de résolution
t, u_Euler = euler(derivee_u, u0, t0, tf, dt)  # résolution par la méthode d'Euler

```

```python
## Cellule 5 : Résolution analytique
def sol_exacte(t):
    """
    Expression analytique de la solution exacte.
    """
    return E*(1-np.exp(-t/tau))
"""
t = np.linspace(tmin,tmax,N)
Renvoie un tableau de N points régulièrement espacés entre tmin (inclus) et tmax (inclus)
"""
t_fixe = np.linspace(t0, tf, 100)       # Variable temporelle contenant 100 valeurs dans [t0,tf]
u_exacte = sol_exacte(t_fixe)           # Calcul de la solution exacte

```

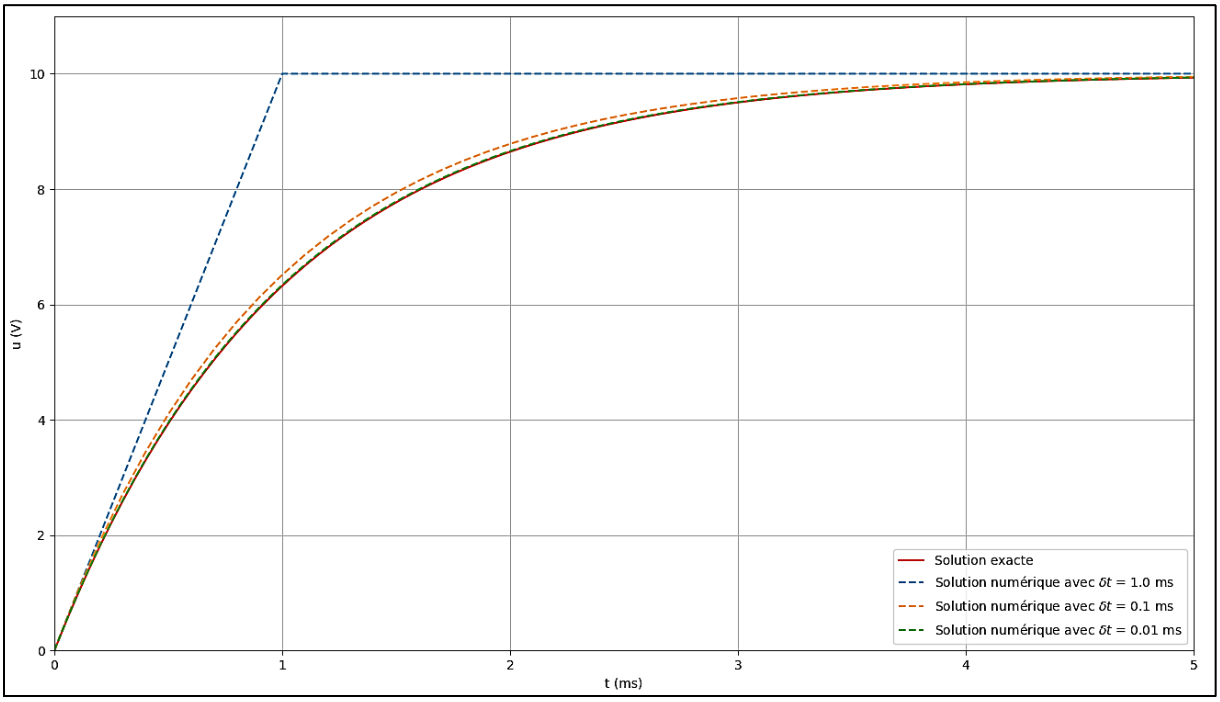
```python
## Cellule 6 : Représentation graphique
plt.figure(figsize=(16,9))          # création d'une fenêtre graphique

plt.plot(t_fixe*1e3, u_exacte, 'r-', label = f"Solution exacte ")   # Graphe temporel de la solution exacte en trait rouge, le temps étant en ms
plt.plot(t*1e3, u_Euler, 'b--', label = r"Solution numérique avec $\delta t = \dfrac{\tau}{10} $"+f" = {dt*1e3} ms") # Graphe temporel de la solution numérique en tirets bleus, le temps étant en ms

# Gestion de l'affichage
plt.xlim(t0,tf*1e3), plt.xlabel('t (ms)')   # habillage de l'axe des abscisses
plt.ylim(0,E+1.), plt.ylabel("u (V)")     # habillage de l'axe des ordonnées
plt.legend(loc = 'lower right')     # affichage de la légende en bas à droite
plt.grid()           # affichage de la grille
plt.show()           # affichage de la figure

```



Commentaire :
La solution numérique est très proche de la solution analytique mais n'est pas rigoureusement identique : ceci est dû au choix du **pas de discrétisation** (pas de résolution numérique).

Influence du pas de discrétisation

```python
## Cellule 7 : Représentation graphique pour plusieurs pas de résolution
plt.figure(figsize=(16,9))          # création d'une nouvelle fenêtre graphique

plt.plot(t_fixe*1e3, u_exacte, 'r-', label = f"Solution exacte ")   # Graphe temporel de la solution exacte en trait rouge, le temps étant en ms

# Boucle de résolutions numériques pour différents pas de résolution et tracé des solutions
for dt in [tau, tau / 10, tau / 100 ]:                     # 3 valeurs du pas de résolution
    t, u_Euler = euler(derivee_u, u0, t0, tf, dt)         # résolution par la méthode d'Euler
    plt.plot(t*1e3, u_Euler, '--', label = r"Solution numérique avec $\delta t $"+f" = {dt*1e3} ms")

# Gestion de l'affichage
plt.xlim(t0,tf*1e3), plt.xlabel('t (ms)')   # habillage de l'axe des abscisses
plt.ylim(0,E+1.), plt.ylabel("u (V)")     # habillage de l'axe des ordonnées
plt.legend(loc = 'lower right')     # affichage de la légende en bas à droite
plt.grid()           # affichage de la grille
plt.show()           # affichage de la figure
```



<u>Commentaire</u> :
Le pas de résolution $\delta t = \tau = 1{,}0$ ms est trop grand et n'est pas adapté à la résolution numérique : la solution obtenue est très éloignée de la solution analytique.

Pour un pas de discrétisation beaucoup plus faible $\delta t = \dfrac{\tau}{100} = 0{,}01$ ms, les solutions numérique et analytique sont identiques.

<u>Conclusion</u> : il faut veiller à prendre un **pas de discrétisation suffisamment petit pour que la solution numérique obtenue soit valable**, notamment quand il n'est pas possible de calculer la solution analytique !