

Principe de résolution numérique des équations différentielles

On cherche à résoudre le problème de Cauchy suivant :

$$\begin{cases} y' = F(y, t) \\ y(a) = y_0 \end{cases}$$

sur un intervalle $[a; b]$. On notera φ son unique solution.

I - Méthode d'Euler (schéma explicite)

On va utiliser la méthode d'Euler avec schéma explicite pour résoudre numériquement l'équation différentielle. Je propose ici le principe :

- On découpe l'intervalle $[a; b]$ en n intervalles de taille $h = \frac{b-a}{n}$.
- On calcule de proche en proche la valeur de la fonction, en utilisant une approximation affine. Supposons que l'on connaît la valeur de $f(t_k)$. Alors, si h est suffisamment petit, on a, avec $t_{k+1} = t_k + h$:

$$\frac{\varphi(t_{k+1}) - \varphi(t_k)}{t_{k+1} - t_k} \sim \varphi'(t_k) = F(\varphi(t_k), t_k)$$

Notons alors $y_k = \varphi(t_k)$. On a donc, à une petite erreur près :

$$y_{k+1} = y_k + h \cdot F(y_k, t_k)$$

Le schéma d'Euler explicite s'écrit donc :

$$\begin{cases} t_0 = a \\ y_0 = y(a) \\ \forall k \in [0, n] \end{cases} \quad y_{k+1} = y_k + h F(y_k, t_k); t_{k+1} = t_k + h$$

Connaissant et $t_0 = a$, $y_0 = y(a)$, on peut donc calculer de proche en proche toutes les valeurs $y_k = \varphi(t_k)$.

II - Un exemple complet traité

Nous testerons cet algorithme grâce au problème de Cauchy :

$$\begin{cases} y' = -2y \\ y(0) = 1 \end{cases}$$

dont l'unique solution est $\varphi(t) = e^{-2t}$, sur $[0; 4]$.

Le but est d'écrire une fonction `euler(F, a, b, y0, n)` qui renverra deux tableaux, `tab_t` et `tab_y` contenant respectivement les abscisses et les ordonnées de la solution de l'équation différentielle approchée par la méthode précédente (on pourra toujours utiliser la méthode `append` pour construire les tableaux).

On commence par importer les modules utiles :

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

On définit ensuite la fonction `F` qui relie y et y'

```
def F(y, t):
    return -2*y
```

Enfin on calcule y pas à pas en utilisant la méthode d'Euler :

```
def euler(F, a, b, y0, n):
    tab_t=[a]
    tab_y=[y0]
    h=(b-a)/n
    t=a
    y=y0
    for i in range(n):
```

```

y=y+h*F1(y,t)
t=t+h
tab_t.append(t)
tab_y.append(y)
return np.array(tab_t), np.array(tab_y)

```

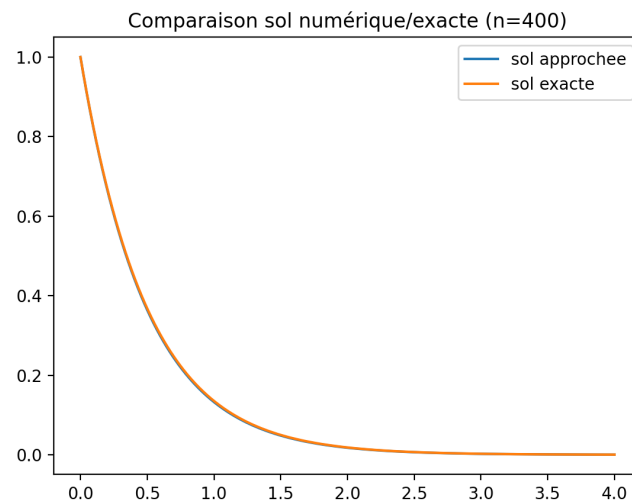
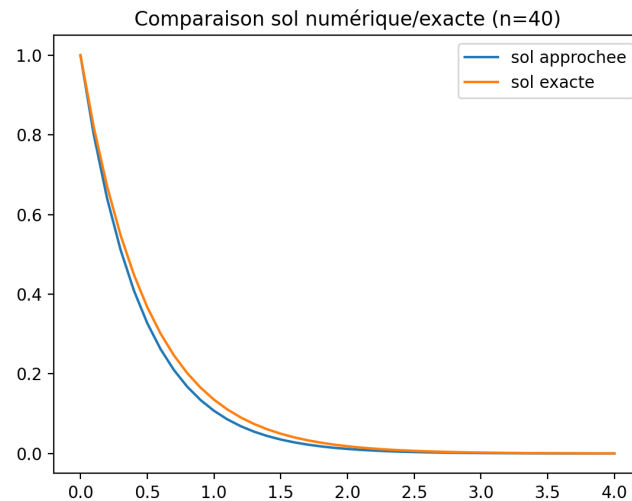
Deuxième étape :

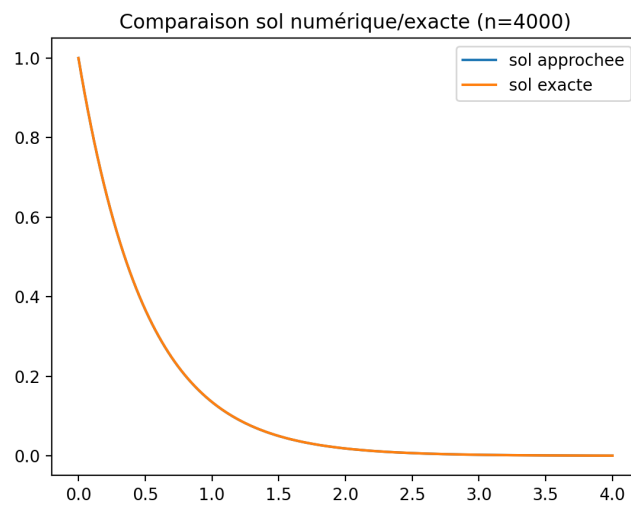
Ecrire une fonction `graphes(F,a,b,y0,n)` qui trace sur la même figure la solution approchée et la solution exacte $t \mapsto e^{-2t}$. On pourra utiliser la fiche d'aide sur "Matplotlib.pyplot" disponible dans le cours de Physique Moodle (via l'ENT).

```

def graphes(F,a,b,y0,n):
    tt,ty=euler(F,a,b,y0,n)
    ty_theo = np.exp(-2*tt)
    plt.plot(tt,ty,label="sol approchee")
    plt.plot(tt,ty_theo,label="sol exacte")
    plt.title('Comparaison sol numérique/exacte (n=40)')
    plt.legend()
    plt.show()

```





Que voit-on pour $n = 40$, $n = 400$ puis $n = 4000$?