

TP - RÉOLUTION NUMÉRIQUE D'UNE ÉQUATION DIFFÉRENTIELLE D'ORDRE 1__MÉTHODE D'EULER

1 La théorie sur la résolution numérique

1.1 L'équation différentielle à résoudre

L'objectif est ici de résoudre une équation différentielle d'ordre 1, connaissant la condition initiale $y(t_0) = y_0$ (problème appelé « problème de Cauchy »). On aura préalablement écrit l'équation différentielle sous la forme :

$$y' = F(y, t)$$

Par exemple : si l'équation différentielle est $2y'(t) + 2y^2(t) - \cos(\omega t) = 2$ alors $F(y, t) = 1 - y^2 + \frac{1}{2}\cos(\omega t)$.

1.2 La méthode d'Euler

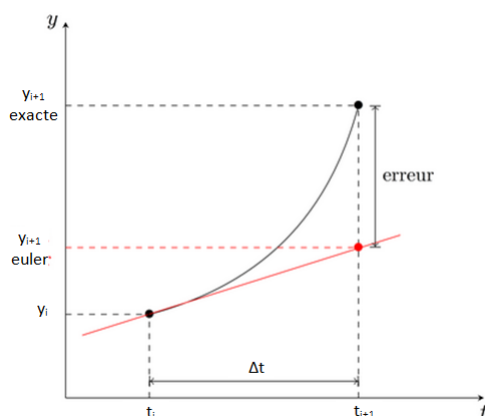
Cette méthode est généralement la première méthode numérique enseignée en raison de sa simplicité mais ses résultats sont malheureusement peu fiables.

Après avoir choisi un pas Δt , on calcule les valeurs approchées de la fonction $y(t)$, solution de l'équation différentielle, à différentes date $t_i = t_0 + i \times \Delta t$, en utilisant le fait que :

$$y'(t_i) = \frac{dy}{dt}(t_i) \approx \frac{y_{i+1} - y_i}{\Delta t}, \text{ ce qui équivaut à : } y_{i+1} \approx y_i + \frac{dy}{dt}(t_i) \times \Delta t$$

On obtient donc :

$$\begin{cases} y(t = t_0) = y_0 \\ y_1 = y(t_1 = t_0 + \Delta t) = y_0 + \frac{dy}{dt}(t_0) \times \Delta t = y_0 + F(y_0, t_0) \times \Delta t \\ y_2 = y(t_2 = t_1 + \Delta t) = y_1 + \frac{dy}{dt}(t_1) \times \Delta t = y_1 + F(y_1, t_1) \times \Delta t \\ \dots \\ y_{i+1} = y(t_{i+1} = t_i + \Delta t) = y_i + \frac{dy}{dt}(t_i) \times \Delta t = y_i + F(y_i, t_i) \times \Delta t \end{cases}$$



Graphiquement, cela revient à déterminer $y_{i+1}(\text{euler})$ en utilisant le coefficient directeur de la tangente à la courbe, à la date t_i , $\frac{dy}{dt}(t_i)$, et donc à commettre l'erreur indiquée sur la figure ci-dessus. Cette erreur peut bien sûr être réduite en diminuant le pas Δt mais cela ne résout pas tous les soucis, comme nous le verrons par la suite.

FIGURE 1 – interprétation graphique de la méthode d'Euler

2 Code Python pour la résolution numérique par la méthode d'Euler

2.1 La fonction F(y,t)

Dans l'exemple du 1.1 et dans le cas où $\omega = \pi \text{ rad} \cdot \text{s}^{-1}$ par exemple, on définit la fonction $F(y, t)$ par :

```
def F(y,t) :
    return 1 - y**2 + 0.5 * np.cos(np.pi * t)
```

2.2 Résolution explicite

Le code suivant permet de résoudre l'équation différentielle $y' = 2y + 1$ sur $[t_{\min}; t_{\max}]$ avec $y(t_{\min}) = 1,5$.

```
T = np.linspace(tmin, tmax, N+1) # N+1 valeurs et N intervalles de longueur (tmax - tmin)/N
y = [1.5]
for i in range ( 1, len(T) ) :
    y = y + (2*y + 1) * ( T[i] - T[i-1] )
```

2.3 La fonction Euler

On propose le code suivant pour définir une fonction, appelée "Euler", réalisant la résolution de l'équation différentielle par la méthode d'Euler : cette fonction renvoie les valeurs $y_0, y_1, \dots, y_N$ rangées dans un tableau numpy. Dans le code suivant, les paramètres G, x_0, a, b et n représentent respectivement la fonction du problème de Cauchy, la condition initiale, l'instant initial, l'instant final et le nombre de sous-intervalles de $[a, b]$.

```
def Euler(G,x0,a,b,n) :
    y = x0
    t = a
    Deltat = (b-a)/n # pas de la subdivision
    L = [y] # on range la première valeur dans une liste
    for i in range(n) :
        y = y + G(y,t) * Deltat # ligne à expliquer ci-après
        t = t + Deltat
        L.append(y) # on range la valeur de y dans une liste
    return np.array(L) # on renvoie un tableau numpy
```

Expliquer l'instruction $y = y + G(y, t) * Deltat$

Dans le cas où la fonction $F(y, t)$ a été définie préalablement et que :

$t_{\min}, t_{\max}, N = 0, 2, 10$

$y_0 = 1$

Ecrire l'instruction pour que la variable Y soit égale au tableau de valeurs $[y_0, y_1, \dots, y_N]$.

2.4 Application

Compléter le code ci-dessous afin de résoudre l'équation différentielle $y' + 2y = 2$ avec $y_0 = 0$.

```
import numpy as np
import matplotlib.pyplot as plt
#
def Euler(G,x0,a,b,n):
    y = x0
    t = a
    Deltat = ..... # pas de la subdivision
    L = [y] # on range la première valeur dans une liste
    for i in range(n):
        y = ...
        t = ...
        L.append(y) # on range la valeur de y dans une liste
    return np.array(L) # on renvoie un tableau numpy
#
def F(y,t) :
    return .....
#
tmin, tmax, N = 0, 10, 100
y0 = .....
#
Y = .....
print(Y)
```

3 Tracé de la solution

3.1 La fonction linspace

Afin de créer le tableau de 11 valeurs $T = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2]$, il est possible...

- d'utiliser l'instruction suivante : $T = \text{np.array}([k * 0.2 \text{ for } k \text{ in range}(11)])$;
- ou d'utiliser la fonction `np.linspace` de numpy en écrivant : $T = \text{np.linspace}(0, 2, 11)$.

En effet, la fonction `np.linspace` de numpy permet de créer un tableau de valeurs régulièrement espacées.

A l'aide de $T = \text{np.linspace}(0, 2, 11)$, l'intervalle $[0, 2]$ a été découpé en 10 sous-intervalles de "longueur" égale au pas $\Delta t = \frac{2-0}{10} = 0,2$

Plus généralement :

$T = \text{np.linspace}(t_{\min}, t_{\max}, N + 1)$ crée un tableau de $N + 1$ valeurs régulièrement espacées avec un pas $\Delta t = \frac{t_{\max} - t_{\min}}{N}$.

Ces $N + 1$ valeurs sont donc égales à :

$$t_{\min}, t_{\min} + 1 \times \Delta t, \dots, t_{\min} + k \times \Delta t, \dots, t_{\min} + N \times \Delta t = t_{\min} + N \times \frac{t_{\max} - t_{\min}}{N} = t_{\max}.$$

Test :

Indiquer l'instruction Python pour créer un tableau de 21 valeurs régulièrement espacées entre -5 et 5 puis tester cette instruction.

3.2 Affichage du graphe

Il est possible de tracer un graphe avec la commande `plt.plot(X,Y)`, X et Y étant des tableaux de valeurs (à 1 colonne) de mêmes longueurs, X contenant les valeurs à porter en abscisses et Y celles à porter en ordonnées.

Compléter le script ci-dessous afin de tracer la courbe obtenue par résolution de l'équation différentielle du 2.3 : solution numérique (label = 'solution numérique') et solution analytique (label = 'solution analytique').

```
tmin , tmax , N = 0 , 10 , 100
tau = ...
T = np.linspace(tmin,tmax, ..... ) #pour découper l'intervalle [tmin,tmax] en N sous-intervalles
plt.figure()
plt.plot(..... , ..... , linestyle='None',marker='+',label='solution numérique')
plt.plot(..... , 1 - np.exp(-T/tau), 'r',label='solution analytique')
plt.xlabel('t')
plt.ylabel('y(t)')
plt.title('Euler')
plt.legend()
plt.show()
```

Compléter ensuite le programme intitulé "Euler_élève.py" afin de trouver la solution du problème de Cauchy :

$$y' + y^2 = 1 \text{ avec } y_0 = 2.$$

3.3 Application à la charge du condensateur

Afin de résoudre numériquement les équations différentielles vues en cours...

- Rappeler l'équation différentielle (en fonction de E et τ) vérifiée par $u_c(t)$ pour un dipôle RC série branché aux bornes d'une source idéale de tension de force électromotrice E .
- Rappeler la solution analytique de cette équation différentielle en fonction de E et τ dans le cas du condensateur initialement déchargé.
- Choisir par exemple $E = 6V$ et $\tau = 2s$ puis modifier le programme précédent afin de résoudre numériquement l'équation différentielle précédente et d'afficher sur un même graphe la courbe obtenue par résolution numérique ainsi que la solution analytique de l'équation différentielle.

Si vous avez le temps : faire de même dans le cas de la décharge du condensateur.