Fiche : Méthode d'Euler vectorielle

Info

1

Principe

- Position du problème :
  Soit X_0 et X_1, deux fonctions définies sur $\mathbb{R}$, t_0, t_f deux réels tels que $t_0<t_f$, et $I = [t_0, t_f]$ l'intervalle de temps de l'étude. Le système pour lequel on cherche une solution approchée est :

$$\begin{cases} X_0'(t) = F(t, X_0, X_1) & (1) \\ X_1'(t) = G(t, X_0, X_1) & (2) \\ X_0(t = t_0) = X_{0,0}, X_1(t = t_0) = X_{1,0} \end{cases}$$

  Les fonctions F et G, définies sur $\mathbb{R}^3$, constituent les *termes de droite* du système. Dans la pratique les fonctions F et G ne dépendront pas nécessairement des trois paramètres t, X_0 et X_1.

- Vectorialisation : On note le vecteur $X = [X_0, X_1]$ et $f = [F, G]$.
- Schéma d'*Euler explicite* :
  Les suites récurrentes (avec pour valeurs initiales $X_{0,0}$ et $X_{1,0}$) constituent le schéma d'Euler explicite :

$$\forall i \in [\![0, N-1]\!], \quad \begin{cases} X_{0,i+1} = X_{0,i} + hF(t_i, X_{0,i}, X_{1,i}) \\ X_{1,i+1} = X_{1,i} + hG(t_i, X_{0,i}, X_{1,i}) \end{cases}$$

  A partir de $X_{0,0}$ et $X_{1,0}$, on peut construire une matrice de valeurs approchées que l'on nommera en Python `valeurs_X` :

$$valeurs\_X = \begin{bmatrix} X_{0,0} & X_{0,1}... & X_{0,N-1} \\ X_{1,0} & X_{1,1}... & X_{1,N-1} \end{bmatrix}$$

  Ainsi qu'un vecteur temps : $t = [t_0 \quad t... \quad t_{N-1}]$

Exemple : cinétique chimique de la réaction A → B

On cherche une solution approchée pour le système d'équations différentielles sur l'intervalle $[0, 5]$ en minutes :

$$\begin{cases} \dfrac{dc_A}{dt} = -kc_A \\ \dfrac{dc_B}{dt} = +kc_A \\ c_{A0} = 1 \text{ mol} \cdot \text{L}^{-1} \text{ et } c_{B0} = 0 \text{ mol} \cdot \text{L}^{-1} \end{cases}$$

Les deux termes de droite des équations différentielles font intervenir les fonctions F et G :

$$\begin{cases} F(c_A) = -kc_A \\ G(c_A) = +kc_A \end{cases}$$

d'où le vecteur $f = [-kc_A, kc_A]$ et la solution à construire :

$$valeurs\_X = \begin{bmatrix} X_{0,0} & X_{0,1}... & X_{0,N-1} \\ X_{1,0} & X_{1,1}... & X_{1,N-1} \end{bmatrix} = \begin{bmatrix} c_{A0} = 1 & c_A(t_1).. & c_A(t_f) \\ c_{B0} = 0 & c_B(t_1).. & c_B(t_f) \end{bmatrix}$$

Ainsi qu'un vecteur temps : $t = [t_0 \quad t... \quad t_{N-1}]$

Solution

- **Étape 1 : définition des constantes, import des bibliothèques**

```
k = 1            #min-1
cA0, cB0 = 1, 0
X0 = [cA0,cB0]  #On initialise le vecteur X
N = 200          #200 points de discrétisation
t0, tf = 0, 5
```

- **Étape 2 : définition de `f` pour les termes de droite**
  Le nouveau schéma numérique est :
  $c_{Ai+1} = c_{Ai} + hF(c_{Ai})$ et
  $c_{Bi+1} = c_{Bi} + hG(c_{Ai})$

```
def f(X):
    cA = X[0]
    cB = X[1]
    return np.array([-k*cA,k*cA])
```

- **Étape 3 : mise en oeuvre de la méthode d'Euler explicite**

```
def Euler_vect(f,t0,tf,X0,N):
    valeurs_X=[X0]
    valeurs_t=[t0]
    X = X0                    #on initialise la matrice
    t = t0                    #on initialise la liste
    h=(tf-t0)/(N-1)
    for i in range(N-1):
        t=t+h
        X = X+h*f(X)          #valable poiur X[0] et X[1]
        valeurs_X.append(X) #on complète la matrice avec le vecteur X
        valeurs_t.append(t) #on complète la liste
    return np.array(valeurs_t), np.array(valeurs_X)
```

- **Étape 4 : appel de `Euler_vect` et tracés**

```
valeurs_t, valeurs_X = Euler_vect(f,t0,tf,X0,N)

plt.title("Résolution numérique: cinétique d'ordre 1, système de 2 équations couplées")
plt.plot(valeurs_t, valeurs_X)
plt.xlabel("Temps (min)")
plt.ylabel("Concentration (mol.L^-1)")
plt.grid(True)
plt.show()
```

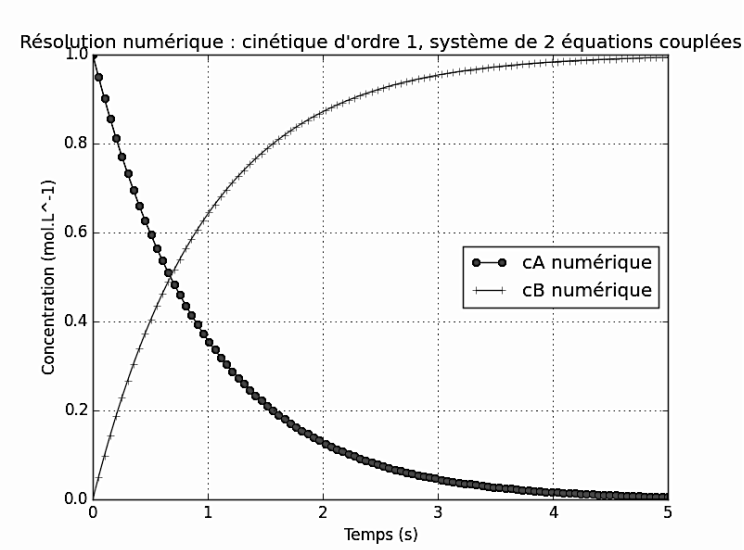

Fig. 01 Réaction A → B ordre 1 : solution numérique des deux équations couplées