Sujet : Physique et Modélisation PC E3A 2016

Quatrième partie
Résolution numérique et traitement d'images

J / Calcul de la déplétion par la méthode d'Euler explicite

J 1. *Fonction* `fIexc(t, maxIexc)`

Il n'y a pas de différences entre τ_{vib} et $tvib$. D'après l'annexe sur le langage **Python**, on peut considérer que le module **numpy** est importé. Par contre je préfère comme toujours l'importation suivante.

```
import numpy as np

def fIexc(t, maxIexc, tvib = 0.001):
    '''renvoie la valeur de l'impulsion de l'excitation.'''
    return( maxIexc * np.exp( - pow(t - 100 * tvib, 2) / \
    ( 2 * pow(10 * tvib, 2) ) ) )
```

J 2. *Valeur des constantes*

La durée de vie τ_{STED} doit être plus grande que τ_{vib} donc $\tau_{STED} = 20 \times \tau_{vib}$ convient ; elle doit être émise après l'impulsion d'excitation ($\mu_{exc} = 100 \times \tau_{vib}$) donc$\mu_{STED} = 150 \times \tau_{vib}$ convient.

J 3. *Code à compléter*

```
tvib = 0.001 ; N = 100 # Nombre de points
tmin = 0.
tmax = 200 * tvib
t = np.linspace(tmin, tmax, N)
h = t[1] - t[0] # ou h = (tmax - tmin) / (N - 1)
```

J 4. *Schéma d'Euler explicite*

$$y_{i+1} = y_i + h \cdot f(t_i,\ y_i)$$

J 5. *Relations de recurrence*

$$\begin{cases} n^*_{1,\ i+1} = n^*_{1,\ i} + h \cdot \left[\sigma_1 \cdot I_{exc}(t_i)(n_{0,\ i} - n^*_{1,\ i}) - k_1 \cdot n^*_{1,\ i}\right] \\ n_{1,\ i+1} = n_{1,\ i} + h \cdot \left[k_1 \cdot n^*_{1,\ i} - \sigma_2 \cdot I_{STED}(t_i)(n_{1,\ i} - n^*_{0,\ i}) - k_2 \cdot n_{1,\ i}\right] \\ n^*_{0,\ i+1} = n^*_{0,\ i} + h \cdot \left[\sigma_2 \cdot I_{STED}(t_i)(n_{1,\ i} - n^*_{0,\ i}) + k_2 \cdot n_{1,\ i} - k_1 \cdot n^*_{0,\ i}\right] \\ n_{0,\ i+1} = n_{0,\ i} + h \cdot \left[-\sigma_1 \cdot I_{exc}(t_i)(n_{0,\ i} - n^*_{1,\ i}) + k_1 \cdot n^*_{0,\ i}\right] \end{cases}$$

Avec les conditions initiales $n_{0,\ 0} = 1$ et $n^*_{0,\ 0} = n_{1,\ 0} = n^*_{1,\ 0} = 0$.

J 6. *Taille des tableaux*

Leur taille est $N + 1$, car on démarre de l'état 0, puis on ajoute N états suivants. C'est cohérent avec la boucle `for` où $i_{max} = N - 1$ et donc $N1_{vib}[i + 1] = N1_{vib}[N]$ à la dernière itération.

J 7. *Code complété*

```
for i in range(N):
    N1vib[i + 1] = ...
    N1[i + 1] = ...
    N0vib[i + 1] = ...
    N0[i + 1] = N0[i] + h * (- sigma1 * fIexc(t[i], maxIexc) * \
    (N0[i] - N1vib[i]) + k1 * N0vib[i])
```

K / Instabilité et schéma implicite

K 1. *Comportement instable de la population*

L'intervalle de temps h choisi est trop grand donc l'équation différentielle n'est pas stable. C'est un problème d'échantillonnage temporel.

On a : $n_{0,\ i+1} = n_{0,\ i} + h \cdot \left[-\sigma_1 \cdot I_{exc}(t_i)(n_{0,\ i} - n^*_{1,\ i}) + k_1 \cdot n^*_{0,\ i}\right]$. La divergence a lieu au maximum de I_{exc}, il faut diminuer le facteur $h \cdot \sigma_1 \cdot I_{exc}(t_i)$. On peut pour cela diminuer le pas h en augmentant N.

K 2. *Condition de convergence*

$q_0(t) = 1 - h \cdot \sigma_1 \cdot I_{exc}(t) = 1 - h \cdot \sigma_1 \cdot \hat{I}_{exc} \cdot e^{-\frac{(t-\mu)^2}{2\cdot\tau^2}}$

Il faut $0 < q_0(t) < 1$. Or la fonction q_0 est majorée strictement par 1. Cette condition est donc vérifiée si $q_0(t) > 0$, c'est à dire si $h \cdot \sigma_1 \cdot \hat{I}_{exc} \cdot e^{-\frac{-(t-\mu)^2}{2\cdot\tau^2}} < 1$.

Or $e^{-\frac{-(t-\mu)^2}{2\cdot\tau^2}}$ est majoré par 1 pour $t = \mu$. La condition de convergence se traduit donc par $h \cdot \sigma_1 \cdot \hat{I}_{exc} < 1$ avec $h = \frac{t_{max}}{N-1}$. Il faut donc $N > 1 + t_{max} \cdot \sigma_1 \cdot \hat{I}_{exc}$.

Avec les données du sujet, en supposant que l'unité de $\hat{I}_{exc}$ à utiliser dans la relation soit le $\mathrm{W \cdot cm^{-2}}$, on trouve $N > 1 + 200 \times 0.001 \cdot \frac{0.001}{100} \cdot \hat{I}_{exc}$.

Il faudrait donc $N = 142$ ou plus points de simulation pour que la suite associée à la population n_0 converge.

K 3. *Schéma d'Euler implicite*

$n_{0,\ i+1} = n_{0,\ i} + h \cdot \left[-\sigma_1 \cdot I_{exc}(t_{i+1})(n_{0,\ i+1} - n^*_{1,\ i+1}) + k_1 \cdot n^*_{0,\ i+1}\right]$
$n_{0,\ i+1} = n_{0,\ i} - h \cdot \sigma_1 \cdot I_{exc}(t_{i+1})(n_{0,\ i+1} - n^*_{1,\ i+1}) + h \cdot k_1 \cdot n^*_{0,\ i+1}$
$n_{0,\ i+1} \cdot (1 + h \cdot \sigma_1 \cdot I_{exc}(t_{i+1})) = n_{0,\ i} + h \cdot \sigma_1 \cdot I_{exc}(t_{i+1}) \cdot n^*_{1,\ i+1} + h \cdot k_1 \cdot n^*_{0,\ i+1}$
Finalement,
$n_{0,\ i+1} = \frac{n_{0,\ i} + h\cdot\sigma_1\cdot I_{exc}(t_{i+1})\cdot n^*_{1,\ i+1} + h\cdot k_1\cdot n^*_{0,\ i+1}}{1 + h\cdot\sigma_1\cdot I_{exc}(t_{i+1})}$

K 4. *Condition de convergence*

Ici, la fonction q_0 est strictement positive, donc la condition de convergence est $q_0(t) < 1, \forall t$. Il faut donc que $1 + h \cdot \sigma_1 \cdot \hat{I}_{exc} \cdot e^{-\frac{-(t-\mu)^2}{2\cdot\tau^2}} > 1$, ce qui est toujours vrai car toutes les constantes sont strictement positives.

Le schéma d'Euler implicite est donc ici inconditionnellement stable.

L / Efficacité STED

L 1. *Méthode des rectangles*

L'aire sous la courbe est approchée par la somme des aires des rectangles calculés à gauche à chaque pas d'abscisse.

On a donc $\int_{t_{min}}^{t_{max}} n_1(t) \cdot \mathrm{d}t \approx h \times \sum_{i=0}^{i=N-1} n_1(t_{min} + i \times h)$, avec $h = \frac{t_{max} - t_{min}}{N-1}$.

L 2. *Méthode d'Euler implicite*

```
def epsilon(N1, N1STED, h, N):
    num, den = 0, 0
    for i in range(N):
        num += (1 - N1STED[i]) * h
        den += N1[i] * h
    return( num / den )
```

L 3. *Classe de complexité*

l'erreur est proportionnelle à h, or $h = \frac{t_{max} - t_{min}}{N-1}$, donc l'erreur est proportionnelle à $\frac{1}{N-1}$, c'est donc un $O(\frac{1}{N})$.

M / Traitement d'images

M 1. *Choix du capteur*

Le capteur est plus sensible aux longueurs d'onde de fluorescence ($\lambda \sim 571$ nm) qu'aux longueurs d'onde STED ($\lambda_{STED} \sim 720$ nm), voir le sujet aux pages 9 et 15. On veut observer les photons de la fluorescence donc le capteur sera plus efficace pour détecter les photons issus de la fluorescence. Mais un filtre sera sans doute nécessaire.

M 2. *Caractéristiques de l'image*

La taille de l'image est $1392 \times 1040 \times 12 = 17\ 372\ 160$ bits, soit 2 171 520 octets.
Le CAN étant de 12 bits, $V_{min} = 0$ et $V_{max} = 2^{12} - 1 = 4095$.

M 3. *Image* `imSrc`

```
imSrc = np.zeros((753, 760))
```

Attention, le premier nombre est celui des lignes, soit 753, avec des indices allant de 0 à 752, et le second est celui des colonnes, soit 760, avec des indices allant de 0 à 759.

M 4. *Inversion des niveaux de gris*

Attention, l'image est à présent en niveaux de gris codés sur 8 bits, donc $V_{max} = 255$ à présent.

```
Vmax = 255
imDest1 = np.zeros((753, 760))
for ligne in range( len(imSrc):
    for colonne in range ( len(imSrc[0]) ):
        imDest1[ligne][colonne] = Vmax - imSrc[ligne][colonne]
```

M 5. *Histogramme des niveaux de gris*

```
hauteur = len(imDest1)
largeur = len(imDest[0])
tabHisto = np.zeros(256) # codage sur 8 bits
for l in range(hauteur):
    for c in range(largeur):
        tabHisto[imDest1[l][c]] += 1
```

M 6. *Correction de l'histogramme des niveaux de gris*

```
scale = 255
valMin, valMax = 212, 248
imDest2 = np.zeros((hauteur, largeur)) # codage sur 8 bits
for l in range(hauteur):
    for c in range(largeur):
        imDest2[l][c] = scale * (imDest1[l][c] - valMin) / (valMax - valMin)
```