DEVOIR SURVEILLÉ D'INFORMATIQUE

Éléments de corrigé

1 Proies – Prédateurs

Les équations de Lotka-Volterra permettent de modéliser le comportement de la reproduction et de la mortalité de proies et de prédateurs sur un territoire. Ce modèle est aussi appelé « proie - prédateur ». Selon ce modèle, les proies se reproduisent. Les prédateurs mangent les proies. Plus il y a de proies, plus les prédateurs se reproduisent et donc le nombre de proies diminue. Lorsque le nombre de proies diminue, le nombre de prédateur diminue, provoquant alors une augmentation de l'effectif des proies ...

Les équations régissant ce modèle sont les suivantes :

$$\begin{cases} u'(t) = u(t)(a - b \cdot v(t)) \\ v'(t) = -v(t)(c - d \cdot u(t)) \end{cases} \quad \text{avec} \quad \begin{cases} u(0) = u_0 \\ v(0) = v_0 \end{cases}$$

Avec :

- $u(t)$ désigne l'effectif de la population des proies ;
- $v(t)$ désigne l'effectif de la population des prédateurs ;
- a : taux de reproduction des proies ;
- b : taux de mortalité des proies ;
- c : taux de mortalité des prédateurs ;
- d : taux de reproduction des prédateurs.

On note h le pas de calcul.

Question 1

Mettre le système différentiel sous la forme d'un schéma d'Euler explicite. On explicitera donc u_{n+1} en fonction de h, u_n, v_n, a et b et v_{n+1} en fonction de h, u_n, v_n, c et d.

Corrigé

On a $\dfrac{du(t)}{dt} \simeq \dfrac{u(t+h)-u(t)}{h}$; donc :

$$\begin{cases} \dfrac{u(t+h)-u(t)}{h} = u(t)(a - b \cdot v(t)) \\ \dfrac{v(t+h)-v(t)}{h} = -v(t)(c - d \cdot u(t)) \end{cases} \Leftarrow \begin{cases} \dfrac{u(n+1)-u(n)}{h} = u(n)(a - b \cdot v(n)) \\ \dfrac{v(n+1)-v(n)}{h} = -v(n)(c - d \cdot u(n)) \end{cases}$$

Au final :

$$\begin{cases} u(n+1) = u(n)h(a - b \cdot v(n)) + u(n) \\ v(n+1) = -v(n)h(c - d \cdot u(n)) + v(n) \end{cases}$$

On donne une suite f_n définie par récurrence. La fonction Python `fonc_f(n)` permet de calculer le nième terme de la suite. On donne la suite g_n définie par récurrence ainsi :

$$g_{n+1} = g_n + h \cdot g_n\left(a - b \cdot f_n\right) \quad \text{avec} \quad g_0 \quad \text{la condition initiale.} \quad a = 0,1 \quad b = 0,01$$

Question 2

Écrire la fonction python `fonc_g` qui permet de calculer le nième terme de la suite g_n.

Corrigé

```python
def fonc_g(n,g0,h):
    a=0.1
    b=0.01
    i=0
    res=g0
    while i<n:
        res = res + h * res (a - b * fonc_f(i))
    return res
```

On donne les fonctions `fonc_u(n)` et `fonc_v(n)` permettant de calculer le nième terme des suites précédemment définies. Sur une durée de 365 jours, on souhaite avoir chaque jour le nombre de proies et de prédateurs.

Question 3

Réaliser en Python la fonction run_ modele qui permet de renvoyer 3 tableaux :

- *tab_j retourne le tableau contenant le numéro du jour ;*
- *tab_ u retourne le tableau contenant le nombre de proies chaque jour ;*
- *tab_ v retourne le tableau contenant le numéro de prédateurs chaque jour.*

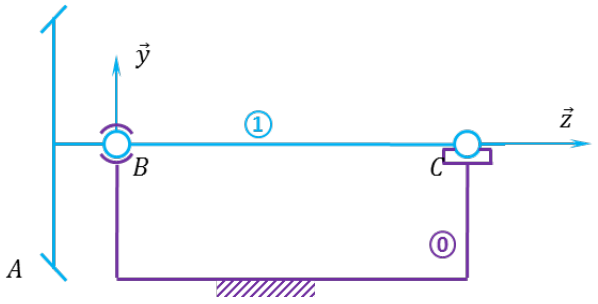
Corrigé

```python
def run_modele()
    tab_j,tab_u,tab_v =[],[],[]
    nb_jour = 365,i=0
    h=1
    while i<=365:
        tab_j.append(i)
        tab_u.append(fonc_u(i))
        tab_v.append(fonc_v(i))
    return tab_j, tab_u, tab_v
```

2 Un peu de statique

Pour terminer son TIPE, un binôme de PTSI doit déterminer les actions mécaniques dans les liaisons en B et C. Une étude préliminaire leur a déjà permis de déterminer les actions mécaniques en A :

$$\{\mathcal{T}(\text{ext} \to 1)\} = \left\{ \begin{array}{cc} 0 & 0 \\ F_y & 0 \\ F_z & 0 \end{array} \right\}_A$$



On donne $\overrightarrow{AB} = l\,\overrightarrow{z} + R\,\overrightarrow{y}$ et $\overrightarrow{BC} = L\,\overrightarrow{z}$.

Les deux acolytes décident (à raison) de se répartir le travail. M. X, décide de se consacrer à l'étude des actions mécaniques. Mlle Y décide de programmer la démarche de résolution en Python. (On remarquera la non galanterie de M. X, se réservant la partie la moins longue du travail).

Question 1 – Question Bonus

Êtes-vous d'accord avec M. X qui a mis en évidence que Y_B, Z_B et Y_C sont solutions du système suivant ?

$$\left(\begin{array}{ccc} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{array} \right) \left(\begin{array}{c} Y_B \\ Z_B \\ Y_C \end{array} \right) = \left(\begin{array}{c} -F_z \\ -F_y \\ -RF_z - lF_y \end{array} \right) \Longleftrightarrow A \cdot X = B$$

Corrigé

On isole 3, qui est soumis à 3 actions mécaniques et applique le PFS en B :

$$\left\{\begin{array}{l} F_z+Z_B=0 \\ F_y+Y_B+Y_C=0 \\ RF_z-lF_y+LY_C=0 \end{array}\right. \Longleftrightarrow \left\{\begin{array}{l} Z_B=-F_z \\ Y_B+Y_C=-F_y \\ LY_C=-RF_z-lF_y \end{array}\right. \Longleftrightarrow \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} Y_B \\ Z_B \\ Y_C \end{pmatrix}=\begin{pmatrix} -F_z \\ -F_y \\ -RF_z-lF_y \end{pmatrix}$$

Pour résoudre ce système, la vaillante informaticienne a décidé d'implémenter la méthode de résolution du pivot de Gauss en Python.

Question 2

Quelles sont les étapes majeures de la méthode du pivot de Gauss ? Comment un pivot est-il défini ?

Les fonctions programmées par Mlle Y sont données ci-dessous.

```python
def fonction1(A,B):
    n=len(A)
    X = [0.] * n
    for i in range(n-1, -1, -1):
        somme=0
        for j in range (i+1,n):
            somme=somme+A[i][j]*X[j]
        X[i]=(B[i][0]-somme)/A[i][i]
        print(X[i])
    return X

def fonction2(A,i):
    n = len(A)
    j = i
    for k in range(i+1, n):
        if abs(A[k][i]) > abs(A[j][i]):
            j = k
    return j

def fonction3(A, i, j, mu):
    nc = len(A[0])
    for k in range(nc):
        A[i][k] = A[i][k] + mu * A[j][k]
    return A

def fonction4(A,i,j):
    A[i][:], A[j][:]= A[j][:], A[i][:]
    return A

def resolution (A, B):
    n = len(A)
    for i in range(n):
        j = ??
        if j > i:
            ??
            ??
        for k in range(i+1, n):
            mu = - A[k][i] / A[i][i]
            ??
            ??
    ??
    return ??
```

Malheureusement, harassée par la fatigue, l'informaticienne en herbe s'endort en n'ayant pas commenté son programme. M. X récupère le programme le lendemain et doit le commenter.

Question 3

Afin de connaître le comportement du programme, utiliser les matrices A et B précédemment définies et donner le résultat des fonctions suivantes :

- *fonction4(A,1,2) ;*
- *fonction3(A,1,2,1) ;*
- *fonction2(A,1).*

Corrigé

- fonction4(A,1,2) :
$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$
- fonction3(A,1,2,1) ;
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$
- fonction2(A,1) : 1

Question 4

Aidez M. X à préciser l'objectif les fonctions 1 à 4.

Corrigé

- Fonction 1 : Phase de remontée
- Fonction 2 : Détermination de la ligne du pivot sur la colonne i
- Fonction 3 : Transvection
- Fonction 4 : Inversion

Question 5

Aidez M. X à compléter les lignes 33, 35, 36, 39, 40, 41 et 42.

Corrigé

```python
def resolution(A, B):
    # Mise sous forme triangulaire
    for i in range(n):
        j = recherche_pivot(A, i)
        if j > i:
            echange_lignes(A, i, j)
            echange_lignes(B, i, j)
        for k in range(i+1, n):
            mu = -A[k][i] / A[i][i]
            transvection_ligne(A, k, i, mu)
            transvection_ligne(B, k, i, mu)
    # Phase de remontée
    X = remontee(A,B)
    return X
```