Projet

Le projet sera rendu sous forme d'un programme python, accompagné d'un document en pdf contenant les réponses aux questions de math et les explications des questions de programmation. Si votre document est manuscrit, écrivez lisiblement et à l'encre noire, laissez des marges et numérotez les pages.
La soutenance aura lieu avec zoom.
Assurez vous d'avoir à cette occasion un ordinateur équipé d'une webcam et d'un micro, et pouvant faire des partages d'écran de votre notebook.

1 Modèle de Lotka-Volterra

En 1926, Volterra proposa une modélisation élémentaire de la dynamique d'une population composée de proies et de leurs prédateurs pour expliquer les proportions fluctuantes de poissons pêchés dans l'Adriatique. En notant $R(\tau)$ (resp. $P(\tau)$) l'effectif des prédateurs (resp. proies) au temps t, le modèle de Volterra est le suivant :

$$\begin{cases} \dfrac{dP}{d\tau} = P(a - bR) \\ \dfrac{dR}{d\tau} = R(cP - d) \end{cases} \tag{1}$$

avec a, b, c, d des constantes positives. Il découle des équations précédentes :

- en l'absence de prédateurs, l'effectif des proies croît indéfiniment de manière Malthusienne (terme aP),
- le taux d'accroissement par tête des proies diminue en proportion du nombre de prédateurs (terme $-bPR$),
- en l'absence de proies, le nombre de prédateurs décroît de façon exponentielle (terme $-dR$),
- la contribution des proies à l'accroissement de la population des prédateurs est proportionnelle au nombre de proies et de prédateurs (terme cPR).

Le produit PR peut s'interpréter en terme d'énergie prise aux proies (bPR) pour aller aux prédateurs (terme cPR). Le principal intérêt de ce modèle, qui a certes ses défauts a été de poser les bonnes questions et d'ouvrir la voie à des modèles plus complexes et plus proches de la réalité.

1) Montrer qu'en posant

$$u(t) = \frac{cP(\tau)}{d}, \quad v(t) = \frac{bR(\tau)}{a}, \quad t = a\tau, \quad \alpha = \frac{d}{a}$$

le système (1) se simplifie en

$$\begin{cases} \dfrac{du}{dt} = u(t)(1 - v(t) \\ \dfrac{dv}{dt} = \alpha v(t)(u(t) - 1). \end{cases} \tag{2}$$

On associe à ce système la donnée initiale

$$\begin{cases} u(0) = u_0, \\ v(0) = v_0, \end{cases} \tag{3}$$

où u_0 et v_0 sont des réels positifs.

2) Etudier l'existence de solutions pour le problème de Cauchy (2),(3). Montrer que si $u_0 > 0$ et $v_0 > 0$ alors $u(t) > 0$ et $v(t) > 0$ pour tout $t > 0$.

3) Montrer que si $(u(t), v(t))$ est une solution de (2) alors la quantité

$$L(u, v) = \alpha u + v - \ln(u^\alpha v) \tag{4}$$

est constante. Montrer que c'est une fonction de Lyapounov pour ce système.

4) Calculer le ou les points d'équilibre du système et étudier leur stabilité.

L'idée de ce projet est de tester différentes méthodes de résolution, dont celle de Scilab, pour les comparer. Divers tests seront proposés pour savoir ce qu'est une bonne méthode numérique. Le but est donc d'approcher la solution $(u(t), v(t))$ pour $t \in [0, T]$, T étant un temps positif. Pour cela, on se donne la suite de temps suivante :

$$0 = t_0 < t_1 < ... < t_N = T$$

et on note $\Delta t = T/(N+1)$. On associe donc à chaque t_n une approximation de $(u(t_n), v(t_n))$, que l'on notera (u_n, v_n). Évidemment, si cette approximation est bien définie, on espère que, quand $\Delta t \to 0$ (ou de manière équivalente quand $N \to +\infty$), la suite $(u_n, v_n)_{n=0,...,N}$ converge vers la solution (u, v) du problème de Cauchy (2),(3). pour $t \in [0, T]$.

5) Définir une fonction python $f : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}^2$ avec comme syntaxe d'appel $f(X, t)$ où les composantes du numpy array X seront u et v.

   Puis à l'aide de la fonction `scipy.integrate.odeint`, tracer la solution du problème de Cauchy (2-3) (la solution obtenue devrait être périodique).

On rappelle que pour une équation différentielle ordinaire, $y' = f(t, y)$, la méthode d'Euler explicite s'écrit

$$y_{n+1} = y_n + f(t_n, y_n).$$

La méthode d'Euler implicite nécessite la résolution d'une équation non-linéaire :

$$y_{n+1} = y_n + f(t_{n+1}, y_{n+1}).$$

On utilisera la fonction `scipy.optimize.fsolve` permettant de résoudre le problème $g(x) = 0$ (ici on choisira bien évidemment pour fonction g la fonction $g(x) = x - f(t_{n+1}, x) - y_n$).

6) Implémenter la méthode d'Euler explicite, la méthode d'Euler implicite, la méthode d'Euler symplectique et la méthode de Runge-Kutta d'ordre 4. On pourra prendre $\alpha = 1, T = 40, N = 20000$, puis tester avec $N = 100000$. Comparer avec la solution donnée par `odeint`.

7) Tracer les solutions approchées dans le plan (u, v) pour chaque méthode employée ci-dessus puis comparer avec la solution donnée par `odeint` obtenue dans la question 5). Vérifier une nouvelle fois le caractère périodique de la solution. Tracer le champ des vitesses de phase, utiliser la fonction `matplotlib.quiver`.

8) Tracer les isovaleurs de la fonction L (fonction `matplotlib.pyplot.contour`).

9) Pour chacune des méthodes implémentées dans la question 6), représenter l'évolution de l'hamiltonien approché $L(u_n, v_n)$ par chaque méthode.

10) Pour chacune des méthodes, calculer l'erreur e_N commise sur l'Hamiltonien au temps $T = 40$ pour N points de discrétisation puis représenter graphiquement cette erreur en fonction de N. En supposant que l'erreur est en $O(N^\beta)$, retrouver la valeur de β pour chaque méthode (on pourra utiliser par exemple `scipy.stats.linregress` en variables logarithmiques)

2 Extension du modèle

On considère la variante du modèle précédent représentant la compétition entre deux espèces P et R

$$\begin{cases} \dfrac{dP}{d\tau} = P(a - \alpha P - bR) \\ \dfrac{dR}{d\tau} = R(cP - d). \end{cases} \tag{5}$$

Seule l'espèce "P" est limitée de manière malthusienne (terme $-\alpha P^2$).

1) Adimensionnaliser le système et déterminer les points d'équilibre.

2) Étudier leur stabilité.

3) Tracer quelques trajectoires dans l'espace des phases.

4) Montrer que le "principe d'exclusion compétitif" est validé par ce modèle quels que soit les paramètres.

5) Quelle interprétation écologique pouvez-vous faire des circonstances menant à l'extinction de l'espèce R ?