Fiche n^0 9. Résolution numérique d'un système prédateurs - proies en langage C

Exercice 1 – **Système de Lotka - Volterra**

Ce système modélise l'évolution de deux populations : celle de proies et celle de prédateurs. Pour plus de détails voir `http://fr.wikipedia.org/wiki/Équations_de_Lotka-Volterra`.

Etant donnés quatre coefficients positifs $(\alpha, \beta, \gamma, \delta)$*, on considère le système suivant*

$$X' = \alpha X - \beta XY,$$

$$Y' = -\gamma Y + \delta XY.$$

a) Comment interprétez vous les différents termes de ces équations, sachant que X *représente la population de proies et* Y *celle de prédateurs ?*

b) Quels sont les états d'équilibre de ce système ?

c) En considérant l'état d'équilibre non-nul $(\overline{X}, \overline{Y})$ *du système, poser* $X = \overline{X} + \Delta X$ *et* $Y = \overline{Y} + \Delta Y$ *et déterminer le système vérifié par* $(\Delta X, \Delta Y)$*.*

d) Pour de petites variations autour de l'équilibre, quelle est la nature de ce système ?

e) Quel est alors l'inconvénient majeur du schéma d'Euler explicite ?

f) Montrez que les schémas d'Euler implicite et de Crank-Nicolson mènent à un système non-linéaire dont on peut trouver les solutions au prix de la résolution d'une équation du second degré.

g) On propose le schéma numérique suivant :

$$\frac{X^{n+1} - X^n}{\Delta t} = \alpha X^n - \frac{\beta}{2} X^n (Y^{n+1} + Y^n)$$

$$\frac{Y^{n+1} - Y^n}{\Delta t} = -\gamma Y^n + \frac{\delta}{2} Y^n (X^{n+1} + X^n)$$

h) Montrez que ce schéma préserve l'état d'équilibre : si $(X^n, Y^n) = (\overline{X}, \overline{Y})$*, alors* $(X^{n+1}, Y^{n+1}) = (\overline{X}, \overline{Y})$*.*

i) Montrez qu'autour de l'équilibre, ce schéma correspond au schéma de Crank-Nicolson pour le système linéarisé.

j) Montrez que, sous une condition que vous préciserez sur le pas de temps, si (X^n, Y^n) *sont positifs, alors il en est de même de* (X^{n+1}, Y^{n+1})*.*

k) Donnez l'expression de (X^{n+1}, Y^{n+1}) *en fonction de* (X^n, Y^n)*.*

Exercice 2 – **Implémentation en Langage C**

Ecrire un programme qui :

-lit dans un fichier le jeu de données : $\alpha, \beta, \gamma, \delta, x(t=0), y(t=0), T, c$*, où* T *est le temps final de la simulation et* c *un réel petit devant* 1*.*

- alloue deux vecteurs `X` *et* `Y` *avec une taille donnée (à choisir judicieusement)*

- implémente une boucle en temps à chaque itération de laquelle :

1) on calcule le pas de temps selon la formule

$$\Delta t = c \min(\frac{1}{\alpha}, \frac{1}{\gamma}, \frac{1}{\beta Y^n}, \frac{1}{\delta X^n})$$

2) on vérifie si on a suffisamment de place mémoire pour calculer `X[n+1]` *et* `Y[n+1]` *et on réalloue si nécessaire. Ceci se fait de la façon suivante*

```
double* temp = NULL; // déclaration hors de la boucle temporelle

// a l'interieur de la boucle et si necessaire, on realloue le vecteur X
// en passant par le vecteur temporaire temp. La nouvelle taille est ntaille_cur

        temp = realloc (xexp, ntaille_cur * sizeof(*X));

        if( temp == NULL ){
          fprintf(stderr,"Reallocation impossible");
          free(X);
          exit(EXIT_FAILURE);
        }
        else{
          X = temp;    // X recoit l'adresse temp
        }
```

3) puis on calcule les valeurs (X^{n+1}, Y^{n+1}) selon le schéma ci-dessus

À la fin du main on écrira dans un fichier les valeurs suivantes (une ligne par pas de temps) :

```
temps écoulé, valeur de X estimée par le schéma,
valeur de Y estimée par le schéma
```

et on visualisera les courbes grâce à `gnuplot`*. On pourra choisir des conditions initiales proches de l'équilibre et on choisira le temps final de simulation de telle sorte à simuler quelques périodes des oscillations.*
