Stratégies optimales d'introduction de coccinelles pour lutter contre des pucerons ravageurs de cultures

Michel de Lara et Aurélie Maure

October 10, 2017

Contents

1 Introduction

Les serres du Sud de la France subissent chaque année l'invasion de colonies de pucerons pendant les cents jours nécessaires pour obtenir le grossissement et la maturation du concombre.

Les dommages causés par ces insectes peuvent se révéler particulièrement importants, entrainant parfois même, en absence de traitement, la destruction du plant. L'utilisation de la coccinelle en tant qu'agent de lutte biologique à grande échelle est une option jusqu'à lors peu utilisée par les agriculteurs et que des biologistes ont décidé d'explorer.

L'introduction de ces coccinelles se fait de façon empirique dans les exploitations. Ici, nous nous plaçons dans une optique de minimisation des coûts sous dynamique biologique et cherchons les stratégies optimales d'introduction des coccinelles. Dans un premier temps, nous étudions le couple coccinelles pucerons quand les coccinelles introduites sont stériles et dans un deuxième temps quand celles-ci ont la capacité de se reproduire.

2 Données biologiques

2.1 Le puceron

Ici, la proie est le puceron A.gossypii. Espèce vivant sur les cucurbitacés, on le retrouve en particulier sur les plants de concombre. Sa présence est particulièrement nocive dans une culture à cause de ses capacités reproductives très rapides. Un ailé pond ses œufs sur une feuille de concombre et, au bout de 2 ou 3 jours, la colonie prend la forme d'une pièce de monnaie qui grossit peu à peu pour atteindre au bout de dix jours un recouvrement total de toute la feuille. Le puceron adulte atteint une masse de 1 mg. La colonisation par la marche peut alors commencer alors que, parallèlement, de nouveaux ailés quittent la feuille pour coloniser d'autres plants. Le puceron a en effet cette capacité d'adaptation au milieu qui lui permet, quand son environnement est saturé en congénaires, de produire des ailés pour étendre la colonie. Ainsi en une quizaine de jours, les pucerons peuvent coloniser toute la serre. Or les ravages causés aux cultures sont très importants : les pucerons sucent la sève de la plante ce qui entraîne son asphyxie. Leurs déjections sont également un poison pour la plante. C'est pourquoi, il nous faut un auxiliaire efficace de lutte biologique, car dès que le nombre de pucerons est d'environ 10 000 par plant, la qualité de celui-ci est dépréciée et la présence d'environ 100 000 d'entre eux entraîne la destruction de la plante.

2.2 La coccinelle

Ici, le prédateur est la coccinelle H.axyridis. Les couleurs de ces coccinelles sont variées mais leurs caractéristiques physiologiques sont identiques. Elle mesure 6 millimètres de long sur 5 millimètres de large et pèse 100 mg. Ces coccinelles sont d'excellents prédateurs pour les espèces d'aphides et donc constituent un outil privilégié dans la lutte biologique contre les pucerons. C'est une espèce qui présente un dévelopement précoce au cours de l'année ainsi qu'une grande capacité reproductrice. De plus, son développement structuré en stades accroît son potentiel à consommer des pucerons. On distingue en effet quatre stades de croissance:

- le stade **œuf** qui dure 3 à 5 jours;

- le stade **larvaire** lui méme subdivisé en quatre sous-stades et dont la durée totale fluctue entre 12 et 14 jours; au cours de ce stade, la coccinelle est une très grande consommatice de pucerons, car sa croissance est directement dépendante de son appétit; la consommation en pucerons est d'autant plus grande que le stade larvaire est avancé;
- le stade **nymphe** d'une durée de 5 à 6 jours et au cours duquel la coccinelle vit sur ses réserves;
- le stade **adulte** où la consommation de pucerons reprend et au cours duquel la coccinelle devient densité dépendante; elle a en effet besoin de 200 à 300 pucerons par jours pour survivre et d'au moins 500 pour décider de se sédentariser sur une feuille infestée.

3 Un modèle proie-prédateur en temps discret

Le système pucerons-coccinelles que nous étudions présente un pas de temps naturel, celui de la journée.

Les variables d'état du modèle sont

- x_t est la biomasse de pucerons en mg ;
- y_t est la biomasse de coccinelles en mg ;

et ses paramètres sont

- r: taux de croissance intrinsèque des proies par jour (sans unité) ;
- K: capacité de soutien du milieu en mg ;
- a: quantité maximale des proies attaquées par jour ;
- k (mg) ; .
- b: pourcentage de proies ingérées par les prédateurs (sans unité) ;
- c: taux de disparition des prédateurs (mortalité et fuite) (sans unité).

$$\begin{cases} x_{t+1} &= x_t + rx_t(1 - \dfrac{x_t}{K}) - \dfrac{ax_t}{k + x_t} y_t \\ y_{t+1} &= \dfrac{bx_t}{k + x_t} y_t - cy_t \end{cases} \tag{1}$$

Question 1 *Trouver les coordonnées $(\bar{x}_i, \bar{y}_i)_{i\in[1;3]}$ des points d'équilibre $P_{i,i\in[1;3]}$ du système pucerons-coccinelles non commandé. Établir la matrice Jacobienne des points P_1 et P_2 et étudier leur stabilité.*

Remarque: On montre qu'une condition suffisante de stabilité du point P_3 est $K(1 - \frac{b}{cr(b-c)}) < \frac{ck}{b-c}$.

Question 2 *Saisir le code suivant dans le fichier* **TP_Lutte_biologique_1.sce** *et simuler des trajectoires partant de* $x = 100\ 000$ *et* $y = 10\ 000$*, soit 100 000 pucerons et 100 coccinelles. Représenter l'évolution quantitative de la biomasse de pucerons et de coccinelle en fonction du temps. Représenter également le portait de phase .*

Question 3 Application numérique*: Sachant que*

- *r=0.4;*
- *a=2;*
- *K=40000;*
- *b=1;*
- *k=100000;*
- *c=0.1;*

interpréter les trajectoires et le portrait de phase.

```
//*****************************************************
// parametres
//*****************************************************

deltat=1
r=0.4*deltat
a=2*deltat
K=40000
b=1*deltat
k=100000
c=0.1*deltat

//*****************************************************
// dynamique
//*****************************************************

function Z=fct(t,Y)
  Z(1)=Y(1)+r .*Y(1)*(1-Y(1)/K)-a .*Y(1)/(k+Y(1)) .*Y(2)
  Z(2)=Y(2)+b .*Y(1) .*Y(2)/(k+Y(1))-c .*Y(2)
endfunction

//*****************************************************
// simulations et graphiques
//*****************************************************
```

```
y0=[100000,10000]';t0=0;t=0:1:500;
//definition des bornes de variation

y=ode('discrete',y0,t0,t,fct);
//calcul des points x_t et y_t pour t dans [0;100]

rectangle=[0,0,t($),100000];//t(£) dernière valeur de t

xset("window",2);
xbasc();
plot2d(t,[y(1,:)',y(2,:)'],style = [1,5])
legends(["pucerons","coccinelles"],[[1;Plein],[5;Plein]],1);
xtitle("Evolution du nombre de coccinelles et de pucerons en fonction du temps")

xset("window",3);
xbasc();
plot2d2(y(1,:),y(2,:))
xtitle("Portrait de phase")
```

4 Un premier problème d'optimisation sans prolifération de la population de coccinelles

4.1 Un premier problème de minimisation des coûts

Dans ce premier modèle, nous considérons un stock de coccinelles pour lesquelles il est impossible de se reproduire. On fixe la valeur de ce stock égale à y_{max}. On considère donc que les seules coccinelles présentes dans le milieu sont celles que nous introduisons, c'est-à-dire que nous leur otons la faculté de se reproduire. L'équation régissant la prolifération de la population de coccinelles disparaît donc de la dynamique (1). La biomasse de pucerons obéit alors à la dynamique

$$x_{t+1} = x_t + rx_t(1 - \frac{x_t}{K}) - \frac{ax_t}{k + x_t}y_t, \text{ où } y_t \text{ est la commande} \tag{2}$$

On note

$$H(x, y) \stackrel{def}{=} x + rx(1 - \frac{x}{K}) - \frac{ax}{k + x}y \tag{3}$$

On considère alors le problème de minimisation des coûts

$$\inf_{y_0,\dots,y_{T-1}} (\sum_{t=0}^{T-1}(C(y_t) + D(x_t)) + D(x_T)) \tag{4}$$

avec la dynamique

$$x_{t+1} = H(x_t, y_t) \tag{5}$$

et sous les contraintes

$$0 \leq y_t \leq y_{max} \tag{6}$$

Ici,

- $C(y)$ est le coût d'introduction de la quantité y de coccinelles par unité de temps, et on n'introduit pas de coccinelles au cours de la dernière étape ; on prendra $C(y) = \alpha + \beta y$,
- $D(x)$ est le coût des dommages dus à une biomasse x de pucerons par unité de temps,
- T est l'horizon.

Nous étudions l'évolution du système pucerons-coccinelles sur une durée de 100 jours, durée de la saison de la culture du concombre, avec une fréquence quotidienne d'introduction de coccinelles. On prendra donc T de l'ordre de 100.

4.2 Résolution par la programmation dynamique

Pour résoudre (4), (5), (6), nous utilisons la méthode de la programmation dynamique. L'équation de Bellman correspondant à notre problème de minimisation des coûts est

$$\begin{cases} V(x,T) &= D(x) \\ V(x,t) &= \min_{0 \leq y \leq y_{max}} [C(y) + D(x) + V(H(x,y), t+1)] \end{cases} \tag{7}$$

Nous minimisons le coût d'introduction des coccinelles et les dommages causés par les pucerons de $t = 0$ à $t = T - 1$. Le coût instantané L est donc

$$\forall t \in [0, T-1], L(x, y, t) = \alpha + \beta y + D(x) \tag{8}$$

$D(x)$ pouvant prendre plusieurs expressions mathématiques selon le modèle que l'on désire tester.

Le coût final Φ est

$$\Phi(x, T) = D(x) \tag{9}$$

La résolution de l'équation (7) nous permet, à chaque pas de temps, de trouver de façon rétrograde le feedback optimal $y^{\#}(x,t)$. $y^{\#}(x,t)$ signifie que la biomasse optimale de coccinelles à introduire au temps t dépend de la biomasse de pucerons.

4.3 Formulation mathématique des fonctions de coût d'introduction et de dommages

Le coût d'introduction des coccinelles au sein la colonie de pucerons est la somme de deux coûts :

- un coût variable βy, produit du nombre de coccinelles par le prix unitaire d'achat de celles-ci ;

- un coût fixe α, qui représente le salaire du technicien chargé de déterminer quels sont les plants les plus infestés dans la serre et donc à traiter prioritairement, mais aussi quelle biomasse de coccinelles il serait préférable d'introduire ; c'est un travail particulièrement long et fastidieux et qui recquiert une excellente acuité et de l'entraînement ; ce coût fixe s'élève à $\frac{1500}{30} = 50$ euros, salaire quotidien du technicien.

L'expression du coût d'introduction est donc très simple car tous les problèmes de modélisation économique de culture, d'élevage et de conditionnement des coccinelles sont résumés dans le prix de vente unitaire d'une coccinelle.

Nous allons considérer deux types de fonction de dommages :

- Des dommages linéaires en pucerons qui sont une première approche un peu grossière du phénomène mais cependant révélatrice de l'impact croissant d'une augmentation du nombre de pucerons sur les cultures :
$$D_l(x) = \delta x \tag{10}$$
- Des dommages non linéaires, qui sont faibles pour de petites quantités de pucerons et qui deviennent "infinis" dès que l'on dépasse un seuil critique de tolérance en pucerons par le plant de concombres :
$$D_{nl}(x) = \frac{\delta}{\gamma - x} \tag{11}$$

4.4 Étude analytique des stratégie optimales : cas d'une dynamique linéaire

La dynamique utilisée est
$$x_{t+1} \overset{defH_1}{=} (1-r)x_t - ay_t \tag{12}$$

Vous procéderez à l'étude analytique complète d'une dynamique linéaire couplée avec une fonction de dommages linéaire (4.4.1) et vous présenterez les premières étapes de résolution d'une dynamique linéaire couplée à avec une fonction de dommages non linéaires (4.4.2). Nous nous limitons à ces deux cas car la résolution analytique de problmès d'optimisation dynamique est en général difficile. Nous en verrons un exemple dans le paragrahe (4.4.2).

4.4.1 Fonction de dommages linéaire $D_l(x)$

$$D_l(x) = \delta x$$

Question 4 *À l'aide de la suite A_t définie par:*
$$\begin{cases} A_{t-1} &= A_t(1-r) + 1 \ , \ \forall t \in [0; T-1] \\ A_T &= 1 \end{cases}$$
donner l'expression de la fonction valeur de Bellman $V(x,t)$ et le feedback asscié $y_t^{\#}$.

Question 5 *Quel type de réponse classique observe-t-on? Commentez le résultat d'un point de vue économique.*

4.4.2 Fonction de dommages non linéaire D_{nl}

$$D_{nl}(x) = \frac{\delta}{\gamma - x}$$

Question 6 *Calculer la fonction valeur jusqu'à l'avant dernière décision. Observer la difficulté de poursuivre le calcul et de trouver une formulation générale de la fonction valeur de Bellman.*

Face aux difficultés analytiques pour déterminer une expression de la fonction valeur de Bellman, nous utilisons des simulations numériques. Pour cela nous allons en premier lieu effectuer la discrétisation du problème.

4.5 Discrétisation du problème

Biomasse de pucerons

Nous considérons une population de pucerons dont la biomasse varie de 0 à 100 000 pucerons. Chaque puceron pesant 1 mg, la population étudiée génère alors jusqu'à 100 000 états possibles pour les pucerons. Or, il est évidemment très coûteux en temps lors de la simulation de créer un nombre aussi élevé d'états. C'est pourquoi nous nous limitons à $p = 500$ états possibles pour la biomasse de pucerons, états pris dans l'ensemble $\{0, h, 2h, ..., 5\ 000h\}$ où h représente la biomasse de 200 pucerons.

Biomasse de coccinelles

De la même façon, la quantité de coccinelles introduites étant nécessairement un multiple de la biomasse d'une coccinelle, nous utilisons un échantillonage variant de 0 à 10 000 mg de coccinelles introduites dans le milieu avec un pas entre chaque valeur de la commande y de 100 mg, valeur de la biomasse d'une coccinelle. Cela représente donc une introduction de 0 à 100 coccinelles.

Discrétisation de la dynamique

La discrétisation du problème (2) conduit à discrétiser la dynamique. Pour cela, nous utilisons des matrices de transition qui vont pour chaque biomasse de puceron dans le milieu nous donner la discrétisation de la biomasse image de la biomasse initiale par la dynamique H. Donc, à chaque image correspond deux biomasses discrétisées X_i et X_{i+1}, telles que $X_i < X < X_{i+1}$. Ces matrices nous permettent d'associer à chaque biomasse de puceron prise entre 0 et 100 000 mg, la biomasse de pucerons discrétisée appartenant à la grille $\{0, h, 2h, ..., 5\ 000h\}$ définie en 4.5. La dynamique H permet de calculer l'image X de la biomasse x considérée initialement. X appartient alors à l'intervalle $[X_i, X_{i+1}]$ formé par deux éléments de la grille.

À X_i et X_{i+1} sont associées des probabilités d'atteignabilité. Ainsi, pour chaque valeur de la commande y,

$$\left\{\begin{array}{lcl} \mathbb{P}(H(x,y)=X_i) & = & p_1 \\ \mathbb{P}(H(x,y)=X_{i+1}) & = & p_2 \\ \mathbb{P}(H(x,y)=X_k) & = & 0 \text{ si } X_k \notin \{X_i; X_{i+1}\} \\ p_1+p_2 & = & 1 \end{array}\right. \tag{13}$$

Alors, pour chaque valeur de la commande est créée une matrice de transition $M^y(x,y)$, indicée par la commande y et telle que $M^y_{ij} = \mathbb{P}(H(x_i,y) = x_j)$. Cette matrice est très creuse et permet le stockage des états possibles de la biomasse de pucerons. Nous avons ainsi discrétisé la dynamique du problème (2)

Matrices de coûts

Le coût instantané est une liste de cardinal le nombre de commandes applicables au système (2), de taille $(cardinal\_etat, horizon)$, $cardinal\_etat$ étant le cardinal de l'espace d'état pucerons. Nous rappelons que le coût instantané L, pour $t \in [0, T-1]$ et le coût final Φ sont

$$\left\{\begin{array}{lcl} L(x,y,t) & = & \alpha + \beta y + D(x) \\ \Phi(x,T) & = & D(x) \end{array}\right. \tag{14}$$

4.6 Résolution numérique par programmation dynamique

Paramètres

Question 7 *Recopier les paramètres suivant dans un fichier* **parametres.sce**

```
//*****************************************************
//parametres.sce
//*****************************************************
// DONNEES

//ymax=10000;//nbmaxcoccinelles
horizon=100;
deltat=0.05;
r=0.4*deltat;
K=40000;
k=100000;
pas=200;
a=2*deltat;
b=1*deltat;
c=0.1*deltat;
alpha=50;
beta=0.52/100;
```

```
delta=8000*102;
//etatdeb=0
//etatfin=100000

//commande= (0:50:ymax)';
commande=[0:100:10000];
// variable en unites physiques (pas des indices)

//etat=etatdeb:pas:etatfin;
etat=[0:200:100000];
```

Fonctions utilisées

Question 8 *Recopier le code suivant dans un fichier* **fonctions.sci** *. Il donne le résultat de la dynamique commandée à chaque couple pucerons-coccinelles, ainsi que les fonctions de dommages et de coût final.*

```
//*****************************************************
//fonctions.sci
//*****************************************************

function image=dynamique_commandee(x,y)
  // dynamique commandee des pucerons
  // ou la commande est directement le nombre de coccinelles
  image=x+r .*x .*(1-x/K)-a .*x .*y ./(k+x)
endfunction

function d=dommages(x)
  d=(deltaprime .*x) .^2;
endfunction;


function c=cout_fin(etat)
  //c=dommages(etat)
  c=deltaprime;
endfunction
```

Discrétisation de la dynamique et passage en matrices de transition

Question 9 *Recopier le code suivant dans un fichier* **macro_discretisation.sci**

Deux fonctions consituent le corps de ce code :

- "**preced_sucess.sci**": qui prend en argument chaque état et son image, vecteur dont tous les éléments ne font pas parti de la grille de discrétisation et retourne les images discrétisées de l'état couplées avec leurs probabilités d'obtention.

- "**discretisation.sci**": En premier lieu, nous précisons que "*discretisation*" est une macro qui a été modifiée en suprimant des boucles qui la constituaient car elle était inefficace pour nos utilisations. La fonction *discretisation* remplissait des matrices carrées. Cependant cette technique n'était pas astucieuse car les matrices ne se remplissaient que de deux éléments par lignes, ce qui générait des boucles pour ne stocker en fait qu'un grand nombre de zéros. Étant donné que nos matrices de transition sont très creuses, nous avons donc utilisé une fonction SCILAB qui crée des "sparse" matrices. Ce sont des matrices à deux colonnes qui n'enregistrent que les positions où les éléments de la matrice sont non nuls et qui leur associent la valeur correspondante. Nous économisons ainsi le temps très coûteux d'une boucle sous SCILAB.
  On récupère les indices des éléments du vecteur image discrétisé à l'aide de "*preced_sucess*" ainsi que les probabilités associées, le code correspondant étant:

  ```
  indices_image_discretisee=predec_sucess(etat,image)
  indices1=indices_image_discretisee(1)
  indices2=indices_image_discretisee(2)
  probabilites=indices_image_discretisee(3)
  ```

  Puis on crée la matrice faisant correspondre pour chaque état l'image discrétisée inférieure x_j, (respectivement supérieure x_{j+1}) ainsi que le vecteur des probabilités associées, *indices_image_discretisee*(3) pour x_j, (respectivement 1-*indices_image_discretisee*(3) pour x_{j+1}, le code correspondant étant

  ```
  ij1(resp2)=(1:prod(size(indices1(resp2))))'
  ij1(resp2)=[ij1(resp2),indices1(resp2)']
  v1(resp2)=probabilites(resp2-probabilites)
  ```

  On poursuit par remplissage de deux sous-matrices de transition, en utilisant la fonction sparse

  ```
  sp1(resp2)=sparse(ij1(resp2),v1(resp2), ...
                    [prod(size(indices1(resp2))),prod(size(indices1(resp2)))])
  ```

  La matrice de transition terminale est donc la somme des deux sous-matrices de transition.

```
//*************************************************
//macro_discretisation.sci
//*************************************************

//macro de discretisation des images
```

```
function indices_image_discretisee=predec_sucess(etat,image)
  //etat = vecteur ordonné
  //image = vecteur
  // indices_image_discretisee = liste
  //indices_image_discretisee(1)(i)=j
  // SSI etat(j) est le prédécesseur de image(i)
  //indices_image_discretisee(2)(i)= j
  // SSI etat(j) est le successeur de image(i)
  //indices_image_discretisee(3)(i)=
  //probabilité d'aller vers le prédécesseur de image(i)

  cardinal_etat=prod(size(etat));
  ind=dsearch(image,etat);
  // image(i) \in [etat(ind(i)), etat(ind(i)+1)[ sauf si
  // ind(i)=0, auquel cas image(i)<etat(1) ou image(i)>etat(£)
  //ind_nd=ind(image_ind);
  image_ind_h=find(image > etat($));
  ind(image_ind_h)=cardinal_etat;
  // ind(i)=0 ssi image(i)<etat(1)
  // ind(i)=n ssi image(i)>etat(£)
  image_ind_m=find(ind > 0 & ind < cardinal_etat);
  // indices du milieu : image(image_ind_m) \subset [etat(1),etat(£)]
  // ind(image_ind_m) \subset ]0,n[
  image_ind_b=find(ind==0);
  // indices du bas : image(image_ind_b) <etat(1)
  image_ind_h=find(ind==cardinal_etat);
  // indices du haut : image(image_ind_h) >etat(£)

  ind1=zeros(image);ind2=ind1;
  proba=zeros(image);

  ind1(image_ind_h)=ind(image_ind_h);
  // envoie les indices pour lesquels l'image de l'etat correspond est
  // >etat(£) vers cardinal_etat, dernier indice de etat
  ind2(image_ind_h)=ind1(image_ind_h);
  // prédécesseur = successeur = etat(£)
  proba(image_ind_h)=ones(image_ind_h);
  // probabilité 1 d'aller vers prédécesseur = successeur

  ind1(image_ind_m)=ind(image_ind_m);
  ind2(image_ind_m)=1+ind(image_ind_m);
  proba(image_ind_m)=(etat(1+ind(image_ind_m))-image(image_ind_m)) ./ ...
```

```
                    (etat(1+ind(image_ind_m))-etat(ind(image_ind_m)));

  ind1(image_ind_b)=ind(image_ind_b)+1;
  // +1 à cause de ind(image_ind_b) composé de 0
  // envoie les indices pour lesquels l'image de l'etat correspond est
  // <etat(1) vers 1, premier indice de etat
  ind2(image_ind_b)=ind1(image_ind_b);
  // prédécesseur = successeur = etat(1)
  proba(image_ind_b)=ones(image_ind_b);
  // probabilité 1 d'aller vers prédécesseur = successeur

  indices_image_discretisee=list();
  indices_image_discretisee(1)=ind1;
  indices_image_discretisee(2)=ind2;
  indices_image_discretisee(3)=proba;
endfunction


function z=egal(x,y)
  z=bool2s(x==y)
endfunction



//NOUVELLE FONCTION DISCRETISATION (SPARSE DISCRETISATION)

function matrice_transition=discretisation(etat,commande,dynamique_commandee)
  matrice_transition=list()
  //etat = vecteur ordonné
  //commande = vecteur
  //dynamique_commandee = fonction(x,u)
  //matrice_transition = liste de matrices de transition
  //  sur les indices de etat
  cardinal_etat=prod(size(etat));
  for l=1:prod(size(commande)) do
    //la liste matrice_transition est indicée par les indices du vecteur commande
    image=dynamique_commandee(etat,commande(l));
    // vecteur des images du vecteur etat
    indices_image_discretisee=predec_sucess(etat,image)
    indices1=indices_image_discretisee(1)
    indices2=indices_image_discretisee(2)
    probabilites=indices_image_discretisee(3)
```

```
    ij1=(1:prod(size(indices1)))'
    ij1=[ij1,indices1']
    v1=probabilites

    sp1=sparse(ij1,v1,[prod(size(indices1)),prod(size(indices1))])



    ij2=(1:prod(size(indices2)))'
    ij2=[ij2,indices2']
    v2=ones(prod(size(probabilites)),1)-probabilites'

    sp2=sparse(ij2,v2,[prod(size(indices2)),prod(size(indices2))])

    matrice_transition(l)=sp1+sp2
  end
endfunction
```

Question 10 *Créer un fichier* **TP_Lutte_biologique_2.sce** *dans lequel vous recopiez le code suivant.*

```
//*****************************************************
//Lutte_biologique_2.sce
//*****************************************************
clear

stacksize(50000000);

// CHARGEMENT DES PARAMETRES PROPES AU MODELE

exec('INFO/parametres.sce');



// CHARGEMENT DES FONCTIONS PROPRES AU MODELE

exec('fonctions.sci');


// CHARGEMENT DES MACROS POUR LA PROGRAMMATION DYNAMIQUE

exec('macro_discretisation.sci');
```

```
// CREATION DE LA LISTE DES MATRICES DE TRANSITION, DES COUTS

matrice_transition=discretisation(etat,commande,dynamique_commandee)

cout_final=cout_fin(etat');
// vecteur

cout_instantane=list();
for i=1:prod(size(commande)) do
  cout_instantane(i)=(alpha+beta*commande(i)) .^2+dommages(etat')*ones(1,horizon);
end;
```

Question 11 *Exécuter le fichier* **Lutte_biologique_2.sce**. *Vérifier que la liste de matrices matrice_transition est bien formé de sparses matrices de transition, c'est à dire de matrices formées de trois colonnes dont la somme des coefficients positifs ou nuls de la dernière colonne, relatifs au même élément de la première colonne, est égale à 1.*

Résolution numérique par programmation dynamique

Recopier dans un fichier **prog_dyn.sci** les codes suivant:

```
//*****************************************************
//prog_dyn.sci
//*****************************************************

//algorithme de r\'esolution r\'etrograde de l\'equation de Bellman.
function [valeur,feedback]=Bell_stoch(matrice_transition,cout_instantane,cout_final)
  // MINIMISATION
  // algorithme de programmation dynamique stochastique
  //     pour une chaîne de Markov sur \{1,2...,cardinal_etat\}
  // matrice_transition = liste à cardinal_commande éléments
  //     composée de matrices de dimension (cardinal_etat,cardinal_etat)
  // cout_instantane = liste à cardinal_commande éléments
  //     composée de matrices de dimension (cardinal_etat,horizon-1)
  // cout_final = vecteur de dimension (cardinal_etat,1)
  // valeur = fonction valeur de Bellman,
  //     matrice de dimension (cardinal_etat,horizon)
  // feedback = commande en feedback sur l'état et le temps
  //     matrice de dimension (cardinal_etat,horizon-1)

  ee=size(cout_instantane(1))
  cardinal_etat=ee(1)
```

```
  cardinal_commande=size(cout_instantane)
  hh=size(cout_instantane(1))
  horizon=1+hh(2)

  valeur=0*ones(cardinal_etat,horizon)
  // tableau ou l'on stocke la fonction de Bellman
  // valeur(:,t) est un vecteur
  // : représente ici le vecteur d'état

  valeur(:,horizon)=cout_final
  // La fonction valeur au temps T est cout_final

  feedback=0*ones(cardinal_etat,horizon-1);
  // tableau ou l'on stocke le feedback u(x,t)

  // Calcul rétrograde de la fonction de Bellman et
  // de la commande optimale à la date 'temp' par l'équation de Bellman
  // On calcule le coût pour la commande choisie connaissant la valeur de la
  // fonction coût à la date suivante pour l'état obtenu

  for temp=horizon:-1:2 do
    // équation rétrograde
    // (attention, pour des questions d'indices, bien finir en :2)
    loc=zeros(cardinal_etat,cardinal_commande);
    // variable locale contenant les valeurs de la fonction valeur de Bellman
    // loc est une matrice
    for j=1:cardinal_commande do
      loc(:,j)=matrice_transition(j)*valeur(:,temp)+cout_instantane(j)(:,temp-1);
    end;

    [mm,jj]=mini(loc,'c')
    // mm est le minimum atteint
    // jj est l'indice de la commande qui réalise le minimum

    valeur(:,temp-1)=mm;
    // coût optimal
    feedback(:,temp-1)=jj;
    // feedback optimal
  end
endfunction
```

```
//Macro de calcul des trajectoires optimales

function z=trajopt(matrice_transition,feedback,cout_instantane,cout_final,etat_initial)
  // z est une liste :
  // z(1) est la trajectoire optimale obtenue partant de etat_initial
  // z(2) est la trajectoire correspondante des commandes optimales
  // z(3) est la trajectoire correspondante des coûts
  // etat_initial = un entier
  // pour le reste, voir la macro Bell_stoch

  ee=size(cout_instantane(1));cardinal_etat=ee(1);
  cardinal_commande=size(cout_instantane);
  hh=size(cout_instantane(1));horizon=1+hh(2);

  z=list();
  x=etat_initial;
  u=[];
  c=[];

  for j=1:horizon-1 do
    u=[u,feedback(x($),j)];
    c=[c,c($)+cout_instantane(u($))(x($),j)];
    mat_trans=full(matrice_transition(u($)));
    x=[x,grand(1,'markov',mat_trans,x($))];
  end
  c=[c,c($)+cout_final(x($))];

  z(1)=x;z(2)=u;z(3)=c;
endfunction
```

Explicitons le problème de la discrétisatio de l'EDP (interpolation linéaire)

Voici l'étape de programmation qui permet le calcul rétrograde des fonctions valeurs de Bellman et des feedbacks optimaux. À chaque pas de temps et pour chaque valeur du vecteur commande, on effectue le calcul de

$$C(y)+D(x)+V(H(x,y),t+1) \approx C(y)+D(x)+pV(H(X_j,y),t+1)+(1-p)V(H(X_{j+1},y),t+1) \tag{15}$$

grâce à

```
for temp=horizon:-1:2 do
  for j=1:cardinal_commande do
    loc(:,j)=matrice_transition(j)*valeur(:,temp)+cout_instantane(j)(:,temp-1)
  end
end
```

Puis on prend le minimum des valeurs à chaque pas de temps, ainsi que l'indice de la commande qui donne le feedback optimal.

```
[mm,jj]=mini(loc,'c')
```

Ces valeurs sont réinjectées à chaque pas de temps dans la récurrence.

Finalisation du fichier TP_Lutte_biologique_2.sce

Question 12 *Recopier, à la suite du code déjà inscrit le code suivant dans le fichier* **TP_Lutte_biologique**

```
exec('prog_dyn.sci');
// RESOLUTION DE L'EQUATION DE LA PROGRAMMATION DYNAMIQUE
[valeur,feedback]=Bell_stoch(matrice_transition,cout_instantane,cout_final)
// TRAJECTOIRES

etat_initial=grand(1,1,'uin',1,prod(size(etat)));
// entier pris uniformement au hasard parmi les indices du vecteur etat

z=trajopt(matrice_transition,feedback,cout_instantane,cout_final,etat_initial)
// calcul des trajectoires optimales (indices)

zz=list();
zz(1)=etat(z(1));
zz(2)=commande(z(2));
zz(3)=z(3);
// trajectoires optimales (unites d'origine)
for l=1:3 do
  xdel(l)
end

xset("window",1);xbasc();
plot2d2(0:(prod(size(zz(1)))-1),zz(1),rect = [0,0,horizon+1,max(etat)]);
xtitle("nombre de pucerons");
xset("window",2);xbasc();
plot2d2(1:prod(size(zz(2))),zz(2),rect = [0,0,horizon,1+max(commande)]);
xtitle("commande");
xset("window",3);xbasc();plot2d2(1:prod(size(zz(3))),zz(3));xtitle("cout cumule")
// tracé des trajectoires optimales

xset("window",4);xbasc();plot(valeur);xtitle("fonction valeur")
//trace de la fonction valeur
```

Nous avons désormais construit toute l'architecture du code nécessaire pour effectuer les simulations.

5 Simulations

5.1 Fonction de dommages linéaire $D_l(x)$

Cas particuliers

Nous allons tester les réactions du programme sur plusieurs cas particuliers. Toutes les valeurs numériques ont été choisies pour que les dommages causés par 2 000 pucerons soient égaux aux coûts d'introduction de la quantité maximale de coccinelles. Les dynamiques utilisées pour simuler les cas particuliers étudiés dans la partie théorique nous conduisent à adapter la valeur du paramètre a. Nous effectuons une mise à l'échelle pour éviter une discrétisation abusive du vecteur image qui prendrait pour la plupart des commandes la valeur 0 à cause de la discrétisation, faussant ainsi les résultats des simulations : cet ajustement est indispensable car des situations aussi particulières que des réponses en "bang-bang" peuvent présenter des marches intermédiaires théoriquement fausses. Les valeurs des paramètres sont

- $\delta = 0.302$
- $a = 10^{-6}$
- $r = 2.10^{-2}$
- état initial= 3000 pucerons.

Question 13 *Simuler les cas suivants :*

- *fonction de dommages linéaire et coûts d'introduction nuls;*
- *fonction de dommages linéaires et dommages nuls;*
- *fonction de dommages linéaires et dynamique non linéaire* $f(x, y) = (r + 1)x - axy$*;*
- *fonction de dommages linéaire et dynamique linéaire* $f(x, y) = (1 - r)x - ay$ *avec un état initial de 80 000 pucerons;*

Pour cela modifier

- *dans* **parametres** *les valeurs de* a*,* δ *et* r
- *dans* **TP_Lutte_biologique_2.sci "TRAJECTOIRES"** *la ligne définissant l'etat_initial;*
- *dans* **fonctions.sci**, *l'expression de la dynamique commandée, des fonctions de coût et de dommages;*

Proposer une interprétation de chaque cas particulier simulés ci-dessus.

Cas général Nous simulons désormais la dynamique $H(x,y) = x + rx(1-\frac{x}{K}) - \frac{ax}{k+x}y$ pour trois états initiaux différents. Nous rapellons que nous sommes toujours dans le cas d' une fonction de dommages linéaires.

Les valeurs des paramètres sont

- $\delta = 0.302$,
- $a = 0.01$,
- $r = 2.10^{-2}$,
- $K = 40\ 000$,
- $k = 100\ 000$
- état initial 1= 3 000 pucerons $\ll K$,
- état initial 2= 40 000 pucerons $= K$,
- état initial 3= 80 000 pucerons $\gg K$

Question 14 *Simuler l'évolution du couple pucerons-coccinelles pour les trois valeurs de l'état initial ci-dessus. Interpréter chacune des simulations obtenues, et tirer une conclusion générale quant aux cas observés.*

5.2 Fonction de dommages non linéaire $D_{nl}(x)$

On définit D_{nl} *par*

$$\begin{cases} D_{nl}(x) &= \frac{\delta}{\gamma - x}, \text{ si } x < 10\ 000 \\ D_{nl}(x) &= 100\ 000, \text{ sinon} \end{cases} \tag{16}$$

Les valeurs des paramètres sont

- $\delta = 0.302$,
- $a = 0.01$,
- $r = 2.10^{-2}$,
- $K = 40\ 000$,
- $k = 100\ 000$

Question 15 *Effectuer les simulations pour les 2 états initiaux :*

- *état initial 1 :* $x = 3000$ *pucerons,*

- *état initial 2 :* $x = 40000$ *pucerons,*

Commenter les graphiques obtenus.

5.3 Coûts d'introduction quadratiques

Reprendre le cas d'une dynamique générale, d'une fonction de dommages non linéaires et d'un coût d'introduction quadratique, $C(y) = (\alpha + \beta y)^2$ *par exemple. Observer le feedback optimal obtenu.*

References

[1] C. Bidot. *Modélisation et commande optimale de l'intéraction Coccinelle/Puceron dans le cadre de la lutte biologique.* Rapport de stage INSA Toulouse, 2000.

[2] M. Duffaut. *Modélisation d'un système proie prédateur dans le cadre de la lutte biologique.* Rapport de stage INSA Toulouse, 2001.

[3] A. Maure *Stratégies optimales pour la lutte biologique : application au couple pucerons-coccinelles* Rapport de stage scientifique Ecole Nationale des Ponts et Chaussées, 2003.

[4] B. D'Andréa Novel & M. Cohen De Lara. *Commande linéaire des systèmes dynamiques.* Masson, 1993

[5] L. Edelstein-Keshet *Mathematical models in biology.* The Random house/Birkhauser mathematics series,1988.

[6] P.Faure. *Analyse numérique, notes d'optimisation.* ellipses,1988.

[7] A. Fera, H. Niknam, F. Kabiri, J-L. Picart, C. De Herce,J. Brun, G. Iperti & L. Lapchin *The use of Harmonia axyridis larvae(coloptera : Coccinellidae) against Macropsiphum rosae(Hemiptera : Sternorrhyncha : Aphididae) on rose bushes.*, Eur.J.Entomol.93:59-67,1996

[8] R. Boll & E. Franco *Protection des cultures sous abri au moyen de la lutte biologique*,Journées du GRAB, 1998