Allocation optimale de ressources en écologie : croître ou se reproduire ? Les questions

Michel DE LARA, Romain CASEY, Anselme LE VAN

October 10, 2017

Contents

1 Allocation optimale en environnement constant (théorie)

1.1 Modèle d'optimisation en environnement constant

Dynamique

En suivant [1], on considère une plante annuelle qui croît et se reproduit selon la dynamique

$$y_t + x_{t+1} = f(x_t)\,, \quad t = 0, ..., T-1 \tag{1}$$

où

- T est le nombre de saisons ($T < +\infty$) ;
- x_t est la biomasse végétative au début de la saison t,
- y_t est la biomasse reproductive produite au cours de la saison t.

Par des considérations biologiques et mathématiques assez générales, voici les hypothèses que nous posons pour la *fonction f de ressources assimilées*, définie sur $[0, +\infty[$:

- f est de classe C^2 sur $]0, +\infty[$;
- $f(0) = 0$ et $f(x) > 0$ pour $x > 0$;
- f est strictement croissante : $f' > 0$;
- f est (strictement) concave : $f'' \leq 0$ ($f'' < 0$).

Mortalité aléatoire

La plante a une durée de vie aléatoire θ de loi géométrique de paramètre β, *probabilité de survie* :

$$\forall t \in \mathbb{N}\,, \quad \mathbb{P}(\theta \geq t) = \beta^t\,. \tag{2}$$

L'évènement $\{\theta = t\}$ signifie que la plante meurt en fin de saison t.

Décisions et stratégies

Si, au début de la saison t, la plante a la biomasse végétative x_t, elle assimile une biomasse totale $f(x_t)$ dont elle peut arbitrer la répartition entre biomasse reproductive y_t et future biomasse végétative x_{t+1}. La décision de la plante correspond au choix de répartition entre biomasse reproductive et future biomasse végétative. On introduit la variable de *décision*

$$u_t = x_{t+1} \in [0, f(x_t)]\,. \tag{3}$$

Une *stratégie* pour la plante est une suite de décisions

$$(u_0, u_1, ..., u_{T-1}) \quad \text{avec} \quad 0 \leq u_0 \leq f(x_0), ..., 0 \leq u_{T-1} \leq f(x_{T-1})\,. \tag{4}$$

Critère

Dans ce modèle d'optimisation, une *stratégie optimale* pour la plante est une stratégie qui rend maximale l'espérance du *fitness* (contribution individuelle aux générations futures), ici la quantité de "rejetons" ou encore le cumul de la biomasse reproductive à la fin de l'année :

$$J(u(.)) = \mathbb{E}(\sum_{t=0}^{(T-1)\wedge\theta} [f(x_t) - u_t]), \quad u(.) = (u_0, u_1, ..., u_{T-1}). \tag{5}$$

L'espérance porte sur la variable aléatoire θ, et un calcul simple montre que l'on peut faire disparaître cette espérance :

$$J(u(.)) = \mathbb{E}(\sum_{t=0}^{(T-1)\wedge\theta} [f(x_t) - u_t]) = \sum_{t=0}^{T-1} \mathbb{E}(\mathbf{1}_{\{\theta\geq t\}}[f(x_t) - u_t]) = \sum_{t=0}^{T-1} \beta^t [f(x_t) - u_t], \tag{6}$$

d'après (2).

Problème d'optimisation

Avec le calcul précédent, ce problème de croissance et reproduction optimales d'une plante se formule alors :

$$\sup_{u_0,u_1,...,u_{T-1}} \sum_{t=0}^{T-1} \beta^t (f(x_t) - u_t) \quad \begin{cases} x_{t+1} = u_t \\ 0 \leq u_t \\ u_t \leq f(x_t) \end{cases} \tag{7}$$

C'est un problème d'optimisation dynamique déterministe, à critère additif, avec

- coût instantané $l(x, u, t) = \beta^t (f(x) - u)$,
- coût final nul.

1.2 Résolution par la programmation dynamique

L'équation de Bellman s'écrit

$$V(x, t) = \max_{0\leq u\leq f(x)} (\beta^t (f(x) - u) + V(u, t + 1)). \tag{8}$$

Si on pose

$$\overline{V}(x, t) := \frac{1}{\beta^t} V(x, t) \tag{9}$$

elle s'écrit également

$$\overline{V}(x, t) = \max_{0\leq u\leq f(x)} (f(x) - u + \beta \overline{V}(u, t + 1)). \tag{10}$$

Les stratégies optimales $u^\sharp$ sont données par

$$u^\sharp(x,t) \in \mathcal{U}^\sharp(x,t) := \arg \max_{0\le u\le f(x)} \left(f(x) - u + \beta \overline{V}(u, t+1)\right). \tag{11}$$

Question 1 *Montrer que $\overline{V}(x, T-1) = f(x)$ et que la dernière commande optimale consiste à investir toute la biomasse végétative en biomasse reproductive à $T-1$, et donc à mourir en T.*

Comme $V(x,T) = 0$ (pas de coût final), on a

$$\overline{V}(x, T-1) = \max_{0\le u\le f(x)} (f(x) - u) = f(x) \quad \text{et} \quad \mathcal{U}^\sharp_{T-1}(x,t) = \{0\}$$

Fonction de croissance linéaire

On suppose ici que

$$f(x) = rx\,. \tag{12}$$

Question 2 *Si $\beta r \le 1$, montrer que, pour $t = 0, ..., T-1$, $\overline{V}(x,t) = rx$ et $\mathcal{U}^\sharp_t(x,t) = 0$. Décrire alors le profil d'une trajectoire optimale.*

Question 3 *Si $\beta r > 1$, montrer que, pour $t = 0, ..., T-1$, $\overline{V}(x,t) = (\beta)^{T-t} rx$ et $\mathcal{U}^\sharp_t(x,t) = rx$. Décrire alors le profil d'une trajectoire optimale.*

Fonction de croissance strictement concave et de pente faible

On suppose ici que

$$\forall x \ge 0\,, \quad f'(x) \le \frac{1}{\beta}\,. \tag{13}$$

Question 4 *Montrer que, pour $t = 0, ..., T-1$, $\overline{V}(x,t) = f(x)$ et $\mathcal{U}^\sharp_t(x,t) = 0$. Décrire alors le profil d'une trajectoire optimale.*

Fonction de croissance strictement concave et de pente forte

On suppose ici que

$$\forall x \ge 0\,, \quad f'(x) > \frac{1}{\beta}\,. \tag{14}$$

Question 5 *Montrer que, pour $t = 0, ..., T-1$, $\mathcal{U}^\sharp_t(x,t) = f(x)$. Décrire alors le profil d'une trajectoire optimale.*

Fonction de croissance strictement concave et de pente modérée

On suppose ici que

$$\exists x_+ > 0\,, \quad \beta f'(x_+) = 1\,. \tag{15}$$

On note alors

$$x_- := f^{-1}(x_+)\,. \tag{16}$$

Question 6 *Montrer qu'une règle optimale de décision au temps $T-2$ est*

- *si la biomasse végétative est inférieure à x_-, alors la biomasse totale (ressources assimilées) sera entièrement transformée en biomasse végétative ;*
- *si la biomasse végétative est supérieure à x_-, alors une partie de la biomasse totale (ressources assimilées) sera transformée en biomasse végétative de manière à atteindre précisement la taille x_+, le reste étant transformé en biomasse reproductive.*

On a

$$\overline{V}(x, T-2) = \max_{0 \le u \le f(x)} (f(x) - u + \beta f(u))\,.$$

D'après les hypothèses sur f, la fonction $u \hookrightarrow -u + \beta f(u)$ est strictement concave, avec un maximum en $u = x_+$. On en déduit aisément que

$$\overline{V}(x, T-2) = \begin{cases} \beta f(f(x)) & \text{si} \quad x \le x_- \\ f(x) - x_+ + \beta f(x_+) & \text{si} \quad x_- \le x \end{cases}$$

Question 7 *Montrer que*

1. *$\overline{V}(x, T-2)$ est continue et continûment dérivable en $x = x_-$, et que $\beta \frac{\partial V}{\partial x}(x_+, T-2) = 1$;*
2. *$\overline{V}(x, T-2)$ est continûment dérivable et que $\frac{\partial \overline{V}}{\partial x}(x, T-2)$ est strictement décroissante.*

En déduire que la règle de décision optimale au temps $t = T-3$, coïncide avec celle au temps $t = T-2$.

En continuant de la sorte, on montre qu'une stratégie optimale pour la plante en feedback sur l'état consiste à suivre la règle suivante pour tout $t = 0, ..., T-2$.

- Si $x \le x_-$, alors toutes les ressources sont dirigées vers la croissance, et la plante ne donne aucun rejeton.
- Si $x \ge x_-$, alors la plante atteint précisément la taille x_+ et transforme sa biomasse restante en biomasse reproductive.

En $t = T-1$, la décision optimale est de transformer toute la biomasse en biomasse reproductive.

2 Comparaison de stratégies en environnement constant (TP Scilab)

On prend

$$f(x,r) = rx^\gamma \quad \text{avec} \quad 0 < \gamma < 1\,. \tag{17}$$

La commande est non plus $u_t \in [0, f(x_t)]$, mais

$$v_t := \frac{u_t}{f(x_t)} \in [0,1] \quad (v_t = 0 \quad \text{si} \quad f(x_t) = 0) \tag{18}$$

2.1 Dynamique de la plante

Ouvrir un fichier `TP_plante_1.sci` et y écrire une fonction Scilab `dyn_plante` représentant la fonction de croissance, en fonction de l'état x (la valeur du paramètre r sera donnée plus tard).

```
function b=dyn_plante(x)
  //b = biomasse totale que la plante la plante peut allouer, sur
  //    une saison, entre biomasse végétative et biomasse reproductive
  //x = biomasse végétative
  b=r*x^{puis}
endfunction
```

Dans le fichier `TP_plante_1.sci` , écrire la fonction `dyn_plante_com`.

```
function b=dyn_plante_com(x,v,s)
  //Fonction définissant la croissance de la plante, commandée par v
  // b = future biomasse végétative
  // x = biomasse végétative
  // v = fraction d'allocation à la croissance (commande)
  // s = facteur de survie (s=1 survie, s=0 mort)
  // si s=0, x transite vers 0
  // si s=1, x transite vers v*dyn_plante(x)
  if v < 0 | v > 1 then
    b='ERREUR : commande au-delà des bornes'
    // | signifie le connecteur logique OU
  else
    if s==0 then
      b=0
      //transition vers 0 si le facteur de survie est nul
    else
      b=v*dyn_plante(x)
    end
  end
endfunction
```

2.2 Fonctions de coût

Dans le fichier `TP_plante_1.sci`, écrire les fonctions de coût instantané et de coût final suivantes.

```
function cout=rejetons(etat,commande,temps)
  cout=(1-commande)*beta^{temps}*dyn_plante(etat);
endfunction

function cout=cout_fin_zero(etat,temps)
  cout=0;
endfunction
```

2.3 Comparaison entre stratégies

Exemples de stratégies

Dans le fichier `TP_plante_1.sci`, écrire les fonctions Scilab suivantes :

- `strategie_M` (M pour mourir) qui représente la stratégie "se reproduire et mourir" : la sortie de `strategie_M` est 0 ;
- `strategie_R` (R pour random) qui représente une stratégie aléatoire : la sortie de `strategie_R` est un réel aléatoire dans $[0, 1]$;
- `strategie_C` (C pour croître) qui représente la stratégie "croître sans se reproduire sauf au dernier pas de temps de commande" : la sortie de `strategie_C` est soit 0 (se reproduire et mourir) soit 1 (croître sans se reproduire).

```
function v=strategie_M(x,t)
  v=0
endfunction

function v=strategie_R(x,t)
  v=rand()
endfunction

function v=strategie_C(x,t)
  if t < T then
    // t=1:T en Scilab correspond à t=0,...,T-1
    v=1
  else
    v=0
  end
endfunction
```

Fonctions Scilab pour le calcul de trajectoires

Dans le fichier `TP_plante_1.sci`, écrire la fonction Scilab `traj_feedback` suivante, qui a pour arguments

- un état initial,
- une fonction Scilab `dynamique_commandee` (sur le modèle de `dyn_plante_com`),
- une fonction Scilab `feedback` (sur le modèle de `strategie_X`),
- un aléa (ici, à deux composantes, aléa de survie et aléa de ressources)

et qui donne, avec ce feedback, les trajectoires correspondantes d'état et de commande.

```
function [x,v]=traj_feedback(etat_initial,dynamique_commandee,feedback,alea)
  // etat_initial
  // dynamique_commandee(x,v,s) : fonction \scilab\ retournant un état
  // feedback(x,t) : fonction \scilab\ retournant une commande
  // alea : suite de 0 ou 1, de longueur horizon
  // x : trajectoire de l'état
  // v : trajectoire de la commande
  horizon=size(alea,2);
  x=etat_initial
  v=[]
  for t=1:horizon do
    // t=0,...,T-1
    v=[v,feedback(x($),t)]
    // x($) est la dernière valeur du vecteur x
    // v est de dimension horizon=T
    x=[x,dynamique_commandee(x($),v($),alea(t))]
    // x est de dimension 1+horizon=1+T
  end
endfunction
```

Dans le fichier `TP_plante_1.sci`, écrire la fonction Scilab `cout_total` suivante, qui a pour arguments

- `trajectoire`, une trajectoire d'état, c'est-à-dire un vecteur de dimension $(1, T+1)$
- `commande`, une trajectoire de commande, c'est-à-dire un vecteur de dimension $(1, T)$
- une fonction Scilab `cout_inst`,
- une fonction Scilab `cout_fin`,

et qui retourne la valeur du critère additif le long des trajectoires d'état et de commande.

```
function cout=cout_total(trajectoire,commande,cout_inst,cout_fin)
  // trajectoire est un vecteur de dimension 1+horizon,
  // commande est un vecteur de commande de dimension horizon
  // cout_inst(etat,commande,temps) est une fonction
  // cout_fin(etat,temps) est une fonction
  // cout est un vecteur de dimension 1+horizon, comprenant les coûts
  //   partiels, sommes cumulées du coût instantané (et du coût final)

  horizon=prod(size(commande))

  if 1+horizon <> prod(size(trajectoire)) then
    cout='ERREUR : dim(trajectoire) différent de 1+dim(commande)'
  else
    cout=[]
    for t=1:horizon do
      // i.e. t=0,...,T-1
      cout=[cout,cout($)+cout_inst(trajectoire(t),commande(t),t)]
      //Stockage du cout instantané
    end
    cout=[cout,cout($)+cout_fin(trajectoire(1+horizon),1+horizon)]
  end
endfunction
```

Fonction de croissance linéaire

Ouvrir un fichier `TP_plante_1.sce` et y recopier les paramètres suivants.

```
// exec('TP_plante_1.sce')
// paramètres
puis=1;//  gamma
beta=0.9,// probabilité de survie
r=1.2;
T=10;
x0=1;
// ATTENTION, si la commande est théoriquement indicée
// par t=0,...,T-1, elle sera indicée par 1:T=1:horizon en Scilab
// l'état sera indicé par 1:(T+1)=1:(1+horizon) en Scilab
```

Question 8 *Calculer les trajectoires d'état et de commande du système en boucle fermée avec les différentes stratégies* ***strategie_X****. On utilisera pour cela la fonction Scilab* ***traj_feedback****.*

Calculer les trajectoires de fitness correspondantes. On utilisera pour cela la fonction Scilab ***cout_total****,*

Tracer les trajectoires d'état, de commande et de fitness à l'aide de la commande Scilab ***plot2d2****.*

Écrire ces instructions dans le fichier `TP_plante_1.sce`*, et l'exécuter par la commande Scilab* `exec('TP_plante_1.sce')`.

```
getf('TP_plante_1.sci');
strategie=list();
strategie(1)=strategie_M;
strategie(2)=strategie_R;
strategie(3)=strategie_C;

correspondance=list();
correspondance(1)=string("mourir");
correspondance(2)=string("aleatoire");
correspondance(3)=string("croitre");

// dans ce qui suit, nous adoptons les notations des arguments
// de la fonction traj_feedback
etat_initial=1;
dynamique=dyn_plante_com;
alea=ceil(beta-rand(1,T));
// T réalisations indépendantes d'une binomiale B(beta,1)
cout_inst=rejetons;
cout_fin=cout_fin_zero;

for i=1:3 do
  [b,v]=traj_feedback(etat_initial,dynamique,strategie(i),alea);
  f=cout_total(b,v,cout_inst,cout_fin);

  xset("window",3*(i-1));xbasc();plot2d2(1:T+1,f,rect = [0,0,T+2,max(f)+1]);
  xtitle("stratégie "+correspondance(i)+" : fitness");

  xset("window",3*(i-1)+1);xbasc();plot2d2(1:T,v,rect = [0,0,T+1,max(v)+1]);
  xtitle("stratégie "+correspondance(i)+" : commande");

  xset("window",3*(i-1)+2);xbasc();plot2d2(1:T+1,b,rect = [0,0,T+2,max(b)+1]);
  xtitle("stratégie "+correspondance(i)+" : état");

  printf("la fitness totale pour la strategie "+correspondance(i)+" est : %"+" f\n",f($)

  halt();
  xdel((3*(i-1)):(3*(i-1)+2));
end
```

Question 9 *Comparer le* fitness *total pour les différentes stratégies.*

Fonction de croissance strictement concave

On prend ici

$$0 < \gamma < 1\,. \tag{19}$$

Question 10 *Donner les expressions analytiques de x_+ et x_-, introduits respectivement aux équations (15) et (16). Écrire en Scilab les formules correspondantes* ***xp*** *et* ***xm****.*

```
// paramètres
puis=0.5;
xp=(beta*r*puis)^{1/(1-puis)};
xm=(xp/r)^{1/puis};
```

Question 11 *Dans le fichier* `TP_plante_1.sci`*, écrire une une fonction Scilab* ***strategie_0*** *(_0 pour optimale), d'arguments un état x et un temps t, de sortie la commande v répondant à la stratégie (feedback) suivante de plante :*

1. *si la biomasse végétative est strictement inférieure à x_-, alors ne pas se reproduire ;*
2. *si la biomasse végétative est supérieure ou égale à x_-, alors au temps suivant elle sera égale à x_+.*

Vérifier, à partir des résultats théoriques vus plus haut, que cette stratégie est optimale.

```
function v=strategie_0(x,t)
  if x < xm then
    v=1
  else
    v=xp/dyn_plante(x)
  end
endfunction
```

Question 12 *Reprendre les questions précédentes.*

```
strategie(4)=strategie_0;
correspondance(4)=string("optimale");
```

Question 13 *Que constate-t-on en faisant varier γ ?*

```
P=0.1:0.2:0.9;
for j=1:prod(size(P)) do
  puis=P(j);
  xp=(beta*r*puis)^{1/(1-puis)};
  xm=(xp/r)^{1/puis};
  x0=xm/10;
  // taille initiale inférieure au seuil xm
```

```
  printf("gamma=%"+" f\n",P(j));
  for i=1:4 do
    [y,v]=traj_feedback(x0,dynamique,strategie(i),alea);
    f=cout_total(y,v,cout_inst,cout_fin);
    printf("la fitness totale pour la strategie "+correspondance(i)+" est : %"+" f\n",f(
  end
end
```

3 Calcul de stratégies optimales en environnement constant (TP Scilab)

On prend ici

$$0 < \gamma < 1\,. \tag{20}$$

3.1 Paramètres

Recopier les paramètres suivants dans un fichier `TP_plante_2.sce`.

```
// exec('TP_plante_2.sce')
T=10;
beta=0.9;
r=1.2;
puis=0.5;
```

3.2 Discrétisation de l'état, de la commande et des coûts

Recopier le code suivant dans le fichier `TP_plante_2.sce` ; il donne la discrétisation de l'état et de la commande, ainsi que le coût instantané et le coût final. On pourra consulter le document *Programmation dynamique, macros générales*.

```
etat=[0:(xm/6):(1.2*xp)];
commande=0:0.05:1;

rejetons=list();
actualisation=cumprod([1,beta*ones(1,T-1)]);
for l=1:prod(size(commande)) do
  rejetons(l)=(1-commande(l))*(r*(etat')^{puis})*actualisation;
  // rejetons(l) est une matrice
end

cout_fin_zero=zeros(etat');
```

On notara qu'on redéfinit, et donc écrase, des fonctions Scilab définies plus haut.

3.3 Discrétisation de la dynamique et passage en matrices de transition

Récupérer les fonctions Scilab `predec_sucess`, `egal` et `discretisation` du document *Programmation dynamique, macros générales* et les copier dans le fichier `TP_plante_2.sci`, ainsi que la fonction suivante.

```
function b=dyn_plante_com(x,v)
  //b = biomasse totale que la plante la plante peut allouer, sur
  //    une saison, entre biomasse végétative et biomasse reproductive
  //x = biomasse végétative
  //v = la commande
  b=v*r*x^{puis}
endfunction
```

Recopier le code suivant dans le fichier `TP_plante_2.sce`.

```
getf('TP_plante_2.sci')

dynamique_commandee=dyn_plante_com;

matrice_transition=discretisation(etat,commande,dynamique_commandee);
```

Question 14 *Vérifier que la liste de matrices* ***`matrice_transition`*** *est bien formée de matrices de transition, c'est-à-dire à coefficients positifs ou nuls, et dont la somme de chaque ligne vaut 1.*

```
cardinal_commande=size(matrice_transition);
for l=1:cardinal_commande do
  sum(matrice_transition(l),"c")'
  mini(matrice_transition(l))
  maxi(matrice_transition(l))
  halt();
end
```

3.4 Résolution numérique par programmation dynamique

Récupérer les fonctions Scilab `Bell_stoch` et `trajopt` du document *Programmation dynamique, macros générales* et les copier dans le fichier `TP_plante_2.sci`.

Recopier ensuite le code suivant dans le fichier `TP_plante_2.sce`.

```
cout_instantane=rejetons;
cout_final=cout_fin_zero;

[valeur,feedback]=Bell_stoch(matrice_transition,cout_instantane,cout_final);
// résoud l'équation de la programmation dynamique

etat_initial=grand(1,1,'uin',2,prod(size(etat)));
// correspond à un état original >= etat(2)
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

xset("window",1);xbasc();plot2d2(1:prod(size(zz(1))),zz(1));xtitle("taille")
xset("window",2);xbasc();plot2d2(1:prod(size(zz(2))),zz(2));xtitle("commande")
xset("window",3);xbasc();plot2d2(1:prod(size(zz(3))),zz(3));xtitle("fitness")
// tracé des trajectoires optimales
```

Question 15 *Examiner les trajectoires, et les comparer avec celles obtenues précédemment.*

4 Allocation optimale en environnement aléatoire (théorie)

4.1 Modèle d'optimisation en environnement aléatoire

On étend le modèle en environnement constant introduit ci-dessus en modèle en environnement stochastique comme suit. On suppose que

- la biomasse totale à allouer au cours d'une saison est à présent fonction des ressources r du milieu, de sorte que la la fonction $x \hookrightarrow f(x)$ est remplacée par une fonction $x \hookrightarrow f(x, r)$ qui possède les mêmes propriétés ;
- au cours de la saison t, les ressources du milieu sont une variable aléatoire R_t prenant ses valeurs dans un ensemble fini $\{r^1, ..., r^m\}$;
- les variables aléatoires R_0,...,R_{T-1} forment une chaîne de Markov : on note π la matrice de terme général $\pi(i, j) = \mathbb{P}(R_{t+1} = r^j | R_t = r^i)$;
- à la saison t, les réalisations des variables aléatoires R_0,...,R_t sont observables par la plante.

À présent, la stratégie v_0,...,v_{T-1} est en feedback sur l'état double (x_t, R_t) : v_t dépend en particulier de R_0,..., R_t qui sont observés.

Dynamique stochastique

La dynamique stochastique de la plante est la suivante : partant de la taille $x \in \mathbb{R}_+$ et de l'environnement r^i, avec la commande $v \in [0, 1]$, à la saison suivante la plante

- meurt, *i.e.* transite vers la taille 0, avec probabilité $1 - \beta$;
- transite vers la taille $vf(x, r^j)$ avec probabilité $\beta\pi(i, j)$.

Commande

Pour des commodités de programmation (bornes fixes sur la commande), on a modifié la commande ($0 \leq v \leq 1$) par rapport à la précédente version ($0 \leq u \leq f(x)$). La commande v est ici la *fraction* de la ressource aléatoire allouée à la biomasse végétative.

Stratégies en feedback sur l'état

Une stratégie optimale pour la plante est une stratégie qui rend maximum l'espérance du *fitness* (contribution individuelle aux générations futures), ici la quantité de "rejetons" ou encore le cumul de la biomasse reproductive à la fin de l'année :

$$J(v(.)) = \mathbb{E}(\sum_{t=0}^{T-1}(1 - v_t)f(x_t, R_t)) \quad \text{avec} \quad v(.) = (v_0, ..., v_{T-1}) \in [0, 1]^T . \tag{21}$$

On peut noter que l'espérance est ici une somme finie.

4.2 Résolution par la programmation dynamique

L'équation de Bellman s'écrit

$$V(x,r,t) = \max_{0\leq v\leq 1} \mathbb{E}\{(1-v)f(x,R_{t+1}) + (1-\beta)V(0,t+1) + \beta V(vf(x,R_{t+1}),t+1) \mid R_t = r\}\,. \tag{22}$$

Question 16 *Montrer par récurrence que $V(0,t) = 0$, de sorte que*

$$V(x,r^i,t) = \max_{0\leq v\leq 1} \sum_{j=1}^{m} \pi(i,j)\{(1-v)f(x,R^j) + \beta V(vf(x,R^j),t+1)\}\,. \tag{23}$$

5 Calcul de stratégies optimales en environnement aléatoire (TP Scilab)

On prend $f(x,r)$ comme en (17), c'est-à-dire

$$f(x,r) = rx^\gamma \quad \text{avec} \quad 0 < \gamma \leq 1\,. \tag{24}$$

Ce dernier chapitre comporte moins d'explications que les précédents. En revanche, le code Scilab est abondamment commenté.

5.1 Discrétisation du problème

Dans ce deuxième modèle, l'état n'est plus comme avant seulement la taille $x \in \mathbb{R}_+$, mais un couple taille – environnement $(x,r) \in \mathbb{R}_+ \times \{r^1, ..., r^m\}$.

Tout d'abord, nous discrétisons l'espace $\mathbb{R}_+$ en $\{x^1, ..., x^n\}$, où x^1 correspond à $x = 0$ (plante morte) et x^n est une taille maximale pour la plante. Nous discrétisons également la fonction $(x,r,v) \hookrightarrow vf(x,r)$ en une fonction

$$F : \{x^1, ..., x^n\} \times \{r^1, ..., r^m\} \times [0,1] \to \{x^1, ..., x^n\}\,. \tag{25}$$

Ensuite, nous exprimons la dynamique stochastique vue plus haut par le biais d'une famille, indicée par la commande v, de probabilités de transition M^v sur l'ensemble produi fini $\{x^1, ..., x^n\} \times \{r^1, ..., r^m\}$.

5.2 Première expression des transitions à l'aide d'hypermatrices

Le code Scilab qui suit exprime ces probabilités de transition M^v par le biais d'hypermatrices à quatre indices. Une hypermatrice est une extension à plus de deux indices des matrices sous Scilab.

Le vecteur `taille` est formé des éléments de la suite $(x^1, ..., x^m)$ des valeurs de la première composante discrétisée de l'état: $taille(i) = x^i$, $i = 1, ..., m$.

5.2.1 Transitions de l'environnement

Le vecteur `env` est formé des éléments de la suite $(r^1, ..., r^m)$ des valeurs possibles prises par les ressources : $env(i) = r^i$, $i = 1, ..., m$.

Dans le fichier `TP_plante_2.sci`, recopier le code suivant.

```
function pi=tr_env_auc_cor_unif(cardinal_alea)
  // aucune corrélation + uniformité
  // les ressources forment une suite de v.a. i.i.d. de loi commune uniforme
  dd=cardinal_alea
  pi=1/dd*ones(dd,dd)
endfunction
```

```
function pi=tr_env_cor_proche_vois(cardinal_alea)
  // corrélation avec les proches voisins
  dd=cardinal_alea
  pi=diag([0.5,1/3*ones(1:(dd-2)),0.5])+diag([0.5,1/3*ones(1:(dd-2))],1)+ ...
     diag([1/3*ones(1:(dd-2)),0.5],-1)
endfunction

function pi=tr_env_cor_crois(cardinal_alea,rho)
  // corrélation avec tendance à croitre
  // rho est l'intensité de la corrélation - rho=0.8
  dd=cardinal_alea
  pi=diag([rho*ones(1:(dd-1)),1])+diag([(1-rho)*ones(1:(dd-1))],1)
endfunction

function pi=tr_env_cor_decrois(cardinal_alea,rho)
  // corrélation avec tendance à décroitre
  // rho est l'intensité de la corrélation - rho=0.8
  dd=cardinal_alea
  pi=diag([1,rho*ones(1:(dd-1))])+diag([(1-rho)*ones(1:(dd-1))],-1)
endfunction

function pi=tr_env_cor_et_chute(cardinal_alea,rho)
  // corrélation avec une probabilité de chute
  // rho est l'intensité de la corrélation - rho=0.9
  dd=cardinal_alea
  pi=diag([rho*ones(1:dd)])
  pi(2:$,1)=1-rho
  pi(1,2)=1-rho
endfunction

function pi=tr_env_cor_et_ascension(cardinal_alea,rho)
  // corrélation avec une probabilité d'ascension
  // rho est l'intensité de la corrélation - rho=0.9
  dd=cardinal_alea
  pi=diag([rho*ones(1:dd)])
  pi(1:$,$)=1-rho
  pi(dd,dd-1)=1-rho
endfunction
```

5.2.2 Transitions de l'état (taille,environnement)

Dans le fichier `TP_plante_2.sci`, recopier le code suivant.

```
function Hypermatrices=constr_HyperM(taille,env,commande,pi,dynamique_commandee)
  // Construction d'une famille de matrices de probabilités de transition sur
  //  \{1,...,n\} X \{1,...,m\} à partir des données du problème :
  // Hypermatrices est une liste d hypermatrices indicées par la commande
  // commande est un vecteur
  // dynamique_commandee(x,r,u) est une fonction à valeurs réelles
  dims=[prod(size(taille)),prod(size(env))];
  ddims=[dims(1),dims(1),dims(2),dims(2)];
  Hypermatrices=list();

  cardinal_commande=prod(size(commande));
  cardinal_taille=prod(size(taille));
  for l=1:cardinal_commande do
    //Hypermat=hypermat([ddims],sparse([],[],[prod(dims),prod(dims)]));
    Hypermat=hypermat([ddims]);
    // pour exprimer simplement les transitions d'un espace produit
    // vers ce même espace

    for y=1:dims(2) do
      image=dynamique_commandee(taille,y,commande(l));
      // vecteur des images du vecteur etat pour r fixé

      indices_image_discretisee=predec_sucess(taille,image);
      indices1=indices_image_discretisee(1);
      indices2=indices_image_discretisee(2);
      probabilites=indices_image_discretisee(3);

      M1=zeros(cardinal_taille,cardinal_taille);
      M2=zeros(M1);
      for i=1:cardinal_taille do
        M1(i,indices1(i))=probabilites(i);
        M1(i,indices2(i))=1-probabilites(i);
      end

      for z=1:dims(2) do
        Hypermat(:,1,y,z)=(1-beta)*pi(y,z);
        //Quand on est vivant (indice>1), on a une probabilité de mort
        // et les transitions de l'environnement sont celles de la matrice pi
        Hypermat(:,:,y,z)=Hypermat(:,:,y,z)+(M1+M2)*beta*pi(y,z);
      end
    end
    // et non pas Hypermat(x,dyn,y,:)=(1-beta)*pi(y,:) end, end
```

```
    // car on peut avoir rempli à l'étape précédente
    // Hypermat(x,dyn,y,:) avec Hypermat(x,1,y,:)=beta*pi.
    Hypermatrices(l)=Hypermat
  end
endfunction
```

5.3 Opérations de conversion pour s'adapter à la fonction Scilab `Bell_stoch`

La fonction Scilab `Bell_stoch` de résolution numérique de l'équation de la programmation dynamique stochastique est écrite pour une chaîne de Markov sur {1,2...,taille_etat} (on obtient en effet un algorithme performant en confondant un état et un indice d'un vecteur).

C'est pourquoi, les hypermatrices de transition à quatre indices seront transformées en matrices ordinaires à deux indices en "déroulant" les indices. Par exemple, une matrice (qui est un cas particulier d'hypermatrice) $\begin{pmatrix} 1 & 3 \\ 7 & 5 \\ 2 & 8 \end{pmatrix}$ sera déroulée en $(1, 7, 2, 3, 5, 8)$.

C'est ainsi que l'espace des états à deux dimensions peut être représenté comme un espace d'état à une dimension. À un couple $(x, r) \in \{x^1, ..., x^n\} \times \{r^1, ..., r^m\}$ est associé l'état $e \in \{1, ..., n \times m\}$. On résout le problème d'optimisation avec cet espace d'état intermédiaire (à une dimension), puis on repasse dans l'espace d'état (taille, environnement) à deux dimensions d'origine.

Question 17 *Dans le fichier* `TP_plante_2.sci`*, recopier le code suivant dans lequel on définit des macros de conversion.*

```
function Matrices=conv_HyperM(Hypermatrices)
  //Matrices est une liste de matrice de transition indicée par
  //la commande
  dd=size(Hypermatrices(1))
  dims=[dd(1),dd(3)]
  //dimensions de l'hypermatrice
  Matrices=list()
  for i=1:prod(size(commande)) do
    Mat=[]
    for j=1:dims(2) do
      if dims(2) <> 1 then
        a=Hypermatrices(i)(:,:,j,:).entries
        b=matrix(a,dims(1),prod(dims))
      else
        b=matrix(Hypermatrices(i)(:,:,j,:),dims(1),prod(dims))
      end
```

```
      Mat=[Mat;b]
      // Transformation en matrice de transition (prod(dims),prod(dims))
      // Cette dernière matrice M est adaptée à la résolution de Bellman
    end
    Matrices(i)=full(Mat)
  end
endfunction


function n=convert1(i,j,dims)
  // i et j sont des coordonnées dans une hypermatrice de dimension dims
  // n est la position de (i,j) dans un vecteur de dimension prod(dims)
  if i > dims(1) | j > dims(2) then
    n='i ou j &gt; dims'
  else
    n=(j-1)*dims(1)+i;
  end
endfunction


function [i,j]=convert2(n,dims)
  // n est la position de (i,j) dans un vecteur de dimension prod(dims)
  // i et j sont des coordonnées dans une hypermatrice de dimension dims
  i=modulo(n,dims(1))
  ind=find(i==0)
  i(ind)=dims(1)
  j=int(n/dims(1))+1
  ind2=find((dims(1))\n==int((dims(1))\n))
  jj=(dims(1))\n
  j(ind2)=jj(ind2)
endfunction


function Hmat=convert2_mat(matrice,dims)
  // matrice est une matrice (ex:feed) dont les éléments proviennent
  //  d'un espace supérieur à 2
  // Hmat est l'hypermatrice correspondant à cette matrice (ex:feed_hyp)
  Hmat=hypermat([dims,T-1])
  n=1:prod(dims)
  [i,j]=convert2(n,dims)
  for t=1:T-1 do
    for indice=1:prod(dims) do
```

```
      Hmat(i(indice),j(indice),t)=matrice(n(indice),t)
    end
  end
endfunction


function cout_instantane=conv_cout(taille,env,commande,cout,horizon)
  //cout_instantane est une liste de cout instantane indicé pour
  // chaque valeur de la commande
  //taille est un vecteur
  //env est un vecteur
  //commande est un vecteur
  // cout est une fonction(x,u,t)
  dims=[prod(size(taille)),prod(size(env))]
  cout_instantane=list()
  for j=1:prod(size(commande)) do
    cout_i=[]
    for t=1:horizon do
      cout_it=hypermat(dims,sparse([],[],[dims(1),dims(2)]));
      for etat1=1:dims(1) do
        for etat2=1:dims(2) do
          cout_it(etat1,etat2)=cout(taille(etat1),env(etat2),commande(j),t)
        end
      end
      cout_it=matrix(cout_it,1,prod(dims))
      // Transformation en vecteur : size(cout_i)=[1,prod(dims)]
      // adapté à la résolution de l'équation de bellman
      cout_i=[cout_i(cout_it)']
    end
    cout_instantane(j)=full(cout_i)
  end
endfunction
```

5.4 Transitions et coûts adaptés à la fonction Scilab `Bell_stoch`

Ouvrir un fichier `TP_plante_3.sce` et y recopier le code suivant.

```
getf('TP_plante_2.sci');


T=10;
beta=0.9;
rr=1.2;
```

```
puis=0.5;
xp=(beta*rr*puis)^{1/(1-puis)};
xm=(xp/rr)^{1/puis};
taille=[0:(xm/3):(1.2*xp)];
//taille=0:2;
env=[0.6,0.8,1,1.2,1.4];
// env=[10:12]
commande=0:0.05:1;
// commande=0:0.5:1;

cardinal_env=prod(size(env));
cardinal_taille=prod(size(taille));

pi=tr_env_auc_cor_unif(cardinal_env);//aucune corrélation + uniformité
//[pi]=tr_env_cor_proche_vois(cardinal_env); //cor avec les proches voisins
//[pi]=tr_env_cor_crois(cardinal_env,rho);
//cor (rho)  avec tendance à croitre
//[pi]=tr_env_cor_decrois(cardinal_env,rho);
//cor (rho) avec tendance à décroitre
//[pi]=tr_env_cor_et_chute(cardinal_env,rho);
//cor (rho) avec une probabilité de chute
//[pi]=tr_env_cor_et_ascension(cardinal_env,rho);
//cor (rho) avec une proba d'ascension
//[pi]=tr_env_cor_crois(cardinal_env,1); //env constant : cor totale

function b=dyn_plante_com(x,r,v)
  //b = biomasse totale que la plante la plante peut allouer, sur
  //    une saison, entre biomasse végétative et biomasse reproductive
  //x = biomasse végétative
  //v = la commande
  b=v*r*x^{puis}
endfunction

dynamique_commandee=dyn_plante_com;

Hypermatrices=constr_HyperM(taille,env,commande,pi,dynamique_commandee);

matrice_transition=conv_HyperM(Hypermatrices);


function ci=rejetons2(taille,env,commande,temps)
  //Fonction définissant la fitness de la plante
```

```
  ci=(1-commande)'*env*taille^(puis)*beta^{temps-1}
  ind=find(taille==1),ci(ind)=0
endfunction

rejetons=conv_cout(taille,env,commande,rejetons2,T);

cout_fin_nul=zeros(cardinal_env*cardinal_taille,1);
//le cout final et nul
```

Question 18 *Vérifier que la liste de matrices* ***matrice_transition*** *est bien formée de matrices de transition, c'est-à-dire à coefficients positifs ou nuls, et dont la somme de chaque ligne vaut 1.*

5.5 Simulation de trajectoires bouclées avec le feedback optimal

Les fonctions nécessaires à la résolution du problème de Bellman et à l'obtention de trajectoires étant à présent écrites dans le fichier `TP_plante_2.sci`, recopier le code suivant dans le fichier `TP_plante_3.sce`.

```
dims=[cardinal_taille,cardinal_env];

stacksize(3000000);

cout_instantane=rejetons;
cout_final=cout_fin_nul;

[valeur,feedback]=Bell_stoch(matrice_transition,cout_instantane,cout_final);

indice_taille_init=grand(1,1,'uin',2,cardinal_taille);
indice_env_init=grand(1,1,'uin',1,cardinal_env);

etat_initial=convert1(indice_taille_init,indice_env_init,dims);

z=trajopt(matrice_transition,feedback,cout_instantane,cout_final,etat_initial);

// calcul des trajectoires optimales (indices)

[indice_taille,indice_env]=convert2(z(1),dims)

x=taille(indice_taille);
e=env(indice_env);
// trajectoires optimales de l'état en unités naturelles
```

```
xset("window",1);xbasc();plot2d2(1:prod(size(x)),x);
xtitle("taille")

xset("window",2);xbasc();plot2d2(1:prod(size(e)),e,rect = [0,0,T+1,max(e)+1]);
xtitle("environnement")

xset("window",3);xbasc();plot2d2(1:prod(size(commande(z(2)))),commande(z(2)));
xtitle("commande")

xset("window",4);xbasc();plot2d2(1:prod(size(z(3))),z(3));
xtitle("fitness")
```

Question 19 *Effectuer différentes simulations. Changer notamment de matrice* ***pi****.*

References

[1] S. Amir and D. Cohen. Optimal reproductive efforts and the timing of reproduction of annual plants in randomly varying environments. *Journal of Theoretical Biology*, 147:17–42, 1990.