Pourriez-vous proposer une solution en Python du problème de biomasse posé en page 161 section 2. Description du modèle. pour maximiser la fonction profit J(x,u) sous les contraintes de l'équation intégrale de type VOLTERA à travers (x, u), variable d'état x (biomasse) et du contrôle u avec la fonction logistique: F(x) = rx(l - x/K) ?

Nous proposons ci‑après une solution numérique en Python qui résout (de façon approximative) le problème de maximisation du profit d’exploitation d’une population « biomasse » décrite par une équation intégrale de type Volterra, dans le cas où la croissance naturelle suit le modèle logistique :

Nous considérons le problème suivant (formulation « discrétisée ») :

* La variable d’état « biomasse » x(t)x(t) vérifie

x(t)=x0+∫0t[r(t−s) F(x(s))−u(s) x(s)] ds,x(t) = x\_0 + \int\_{0}^{t} \Bigl[r(t-s) \, F(x(s)) - u(s)\, x(s)\Bigr]\,ds,

où

* x0>0x\_0>0 est le stock initial,
* F(x)=rlog x(1−xK)F(x) = r\_{\text{log}}\, x\Bigl(1-\frac{x}{K}\Bigr) (la croissance logistique) avec rlog>0r\_{\text{log}}>0 et K>0K>0 le niveau de saturation,
* la fonction « retenue » est choisie par exemple r(t−s)=exp⁡(−p(t−s))r(t-s)=\exp\bigl(-p(t-s)\bigr) avec p>0p>0 (responsable de la « dissipation » cumulée).
* Le contrôle u(t)u(t) (représentant l’effort d’extraction ou de récolte) appartient à l’intervalle [0,E][0, E].
* L’objectif est de maximiser le profit total actualisé

J(x,u)=∫0Te−δt u(t) x(t) dt,J(x,u) = \int\_{0}^{T} e^{-\delta t}\, u(t)\, x(t)\,dt,

où T>0T>0 est l’horizon de planification et δ>0\delta>0 le taux d’actualisation.

Pour résoudre ce problème nous procédons de la manière suivante :

1. Nous discrétisons l’intervalle [0,T][0,T] en NN sous-intervalles de pas Δt=T/N\Delta t = T/N.
2. La formulation intégrale se traduit alors par le système de récurrence (pour i=1,…,Ni=1,\dots,N) :

xi=x0+Δt ∑j=0i−1[exp⁡(−p (ti−tj)) rlog xj(1−xjK)−uj xj],x\_i = x\_0 + \Delta t\,\sum\_{j=0}^{i-1} \Bigl[\exp\bigl(-p\,(t\_i-t\_j)\bigr)\, r\_{\text{log}}\, x\_j\Bigl(1-\frac{x\_j}{K}\Bigr) - u\_j\,x\_j\Bigr],

avec ti=iΔtt\_i = i\Delta t et x0x\_0 donné.

1. L’objectif discret devient :

J(u)≈∑i=0N−1e−δti ui xi Δt.J(u) \approx \sum\_{i=0}^{N-1} e^{-\delta t\_i}\, u\_i\, x\_i\,\Delta t.

(Nous cherchons à maximiser JJ; pour utiliser un solveur de minimisation, nous maximisons en minimisant l’opposé.)

1. Le problème est à résoudre avec ui∈[0,E]u\_i \in [0,E] pour i=0,…,N−1i=0,\dots,N-1. Pour que l’état reste « raisonnable » nous ajouterons une pénalité si l’un des xix\_i tombe en-dessous de 0.

Ci‑après le code complet en Python qui met en oeuvre cette approche (la discrétisation de la contrainte dynamique de type Volterra ainsi que la détermination du contrôle optimal par un solveur non linéaire) :

python

import numpy as np

from scipy.optimize import minimize

import matplotlib.pyplot as plt

def simulate\_dynamics(u, params):

"""

Retourne la trajectoire x et le vecteur de temps t

obtenus à partir des contrôles u donnés et des paramètres.

Les paramètres contenus dans 'params' sont :

- T : horizon de planification (float)

- N : nombre de pas de discrétisation (int)

- x0 : biomasse initiale (float)

- r\_growth : taux de croissance logistique r\_log (float)

- K : capacité de saturation (float)

- p : paramètre de dissipation (float) dans r(t-s)=exp(-p\*(t-s))

"""

T = params['T']

N = params['N']

dt = T / N

t = np.linspace(0, T, N+1)

x = np.zeros(N+1)

x[0] = params['x0']

# Récurrence : pour i>=1

for i in range(1, N+1):

summation = 0.0

for j in range(i):

# Calcul de la contribution de l'intervalle [t\_j, t\_j+1]

growth = params['r\_growth'] \* x[j] \* (1 - x[j]/params['K'])

retention = np.exp(-params['p'] \* (t[i] - t[j]))

summation += (retention \* growth - u[j] \* x[j]) \* dt

x[i] = params['x0'] + summation

return x, t

def objective(u, params):

"""

Fonction objectif à minimiser (négatif du profit)

Le profit est défini par l'intégrale discrétisée

J = ∫0^T e^{-δt} u(t)\*x(t) dt.

Si la trajectoire x présente des valeurs négatives, une pénalité importante est ajoutée.

"""

x, t = simulate\_dynamics(u, params)

dt = params['T'] / params['N']

profit = 0.0

for i in range(params['N']):

profit += np.exp(-params['delta'] \* t[i]) \* u[i] \* x[i] \* dt

# Pénalité si la biomasse devient négative

if np.any(x < 0):

penalty = 1e6 \* np.sum(np.abs(x[x < 0]))

profit -= penalty

# Nous maximisons le profit, d'où l'opposé

return -profit

def main():

# Définition des paramètres du modèle

params = {

'T': 10.0, # horizon de planification

'N': 100, # nombre de points de discrétisation

'x0': 1.0, # biomasse initiale

'r\_growth': 0.5, # taux de croissance logistique

'K': 10.0, # capacité de saturation

'p': 0.1, # paramètre de dissipation (dans r(t-s)=exp(-p\*(t-s)))

'delta': 0.05, # taux d'actualisation

'E': 1.0, # effort maximal admissible (u in [0,1])

}

N = params['N']

# Initialisation du contrôle (par exemple, effort constant à la moitié de la borne maximale)

u0 = np.full(N, params['E'] / 2)

# Définir les bornes pour chaque contrôle : u[i] ∈ [0, E]

bounds = [(0, params['E'])] \* N

# On utilise la méthode SLSQP du solveur scipy.optimize.minimize

res = minimize(objective, u0, args=(params,), bounds=bounds, method='SLSQP',

options={'maxiter': 1000, 'disp': True})

# Récupération du contrôle optimal et calcul de la trajectoire correspondante

u\_opt = res.x

x\_opt, t = simulate\_dynamics(u\_opt, params)

dt = params['T'] / params['N']

profit = sum(np.exp(-params['delta'] \* t[i]) \* u\_opt[i] \* x\_opt[i] \* dt for i in range(N))

# Affichage des résultats

plt.figure(figsize=(8,6))

plt.subplot(2,1,1)

plt.plot(t, x\_opt, 'b-', label='Biomasse x(t)')

plt.xlabel('Temps t')

plt.ylabel('Biomasse')

plt.legend()

plt.subplot(2,1,2)

# Pour afficher u (défini sur chaque intervalle, on affecte son graphique aux points t[0:N])

plt.step(t[:-1], u\_opt, 'r-', where='post', label='Contrôle u(t)')

plt.xlabel('Temps t')

plt.ylabel('Effort de récolte u(t)')

plt.legend()

plt.tight\_layout()

plt.show()

print("Profit optimal =", profit)

print("Contrôle optimal u =", u\_opt)

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Explications complémentaires**

1. **Discrétisation de l’équation intégrale :** Pour chaque pas de temps tit\_i avec i=0,…,Ni=0, \dots, N, nous approchons l’intégrale par une somme (méthode du rectangle). Ainsi, connaissant x0x\_0 et avec un contrôle uu défini sur chaque intervalle, la trajectoire xx se calcule récursivement.
2. **Fonction objectif :** Le profit actualisé est évalué par la somme discrète

J(u)≈∑i=0N−1e−δti ui xi Δt.J(u) \approx \sum\_{i=0}^{N-1} e^{-\delta t\_i}\,u\_i\,x\_i\,\Delta t.

Comme le solveur minimise, nous retournons −J(u)-J(u).

1. **Contrainte sur** uu **:** Les bornes imposent ui∈[0,E]u\_i \in [0,E]. En cas de problème (par exemple si xix\_i devenait négatif, ce qui ne serait pas réaliste), une pénalité très élevée est ajoutée.
2. **Solveur :** Nous utilisons le solveur SLSQP (méthode des moindres carrés séquentiels) qui gère naturellement les contraintes par bornes.

Ce script fournit un point de départ pour explorer la solution optimale du problème de biomasse sous la modélisation en équation intégrale de type Volterra avec croissance logistique et effet de dissipation ; il conviendra éventuellement d’ajuster les paramètres et/ou la discrétisation pour des applications concrètes.

N’hésitez pas à poser des questions complémentaires ou à proposer des améliorations complémentaires dans l’approche numérique.

J'ai réalisé plusieurs simulations avec le code Python, mais la biomasse décroit systématiquement les 1ères années en atteignant des valeurs négatives, ce qui semble être irréaliste ?

Cela peut provenir de plusieurs facteurs, notamment du schéma de discrétisation et des paramètres choisis. Voici quelques pistes à explorer :

1. **Le schéma d’intégration (méthode du rectangle) :** En discrétisant l’intégrale par une somme de rectangles simples avec un pas Δt\Delta t peut parfois introduire des erreurs importantes, surtout si Δt\Delta t est trop grand. Ces erreurs peuvent alors rendre la dynamique numérique instable (la biomasse pouvant devenir négative) même si la solution théorique reste positive. Une solution consiste à réduire le pas de temps ou à utiliser un schéma d’intégration plus stable (par exemple une méthode implicite ou une méthode de Runge–Kutta adaptée aux équations avec des contraintes de positivité).
2. **Paramètres du modèle :**
   * Si le contrôle u(t)u(t) (la récolte) ou les paramètres choisis (le taux de croissance logistique rlogr\_{\text{log}}, la capacité KK, ou encore le paramètre pp dans le terme de rétention r(t−s)=exp⁡(−p(t−s))r(t-s)=\exp(-p(t-s))) sont configurés de manière à ce que l’extraction soit trop agressive, il se peut que le terme négatif −u(t)x(t)-u(t)x(t) domine la croissance naturelle. Par conséquent, pour certaines valeurs, la somme en fermeture des contributions croisées devient négative.
   * Vérifiez que u(t)u(t) reste dans des plages raisonnables et que l’équilibre entre la croissance F(x)=rlog x (1−xK)F(x)=r\_{\text{log}}\,x\,(1-\frac{x}{K}) et l’extraction est bien calibré pour obtenir un comportement réaliste (aucune décroissance trop rapide).
3. **Contraintes sur l’état :** Dans le problème réel, la biomasse doit être non négative. Il peut être judicieux d’ajouter explicitement une contrainte dans votre optimisation pour forcer x(t)≥0x(t)\ge0 (même si cela complique un peu le problème) ou d’utiliser une pénalité plus forte lorsque x(t)x(t) tend vers des valeurs négatives.
4. **Approche de post‑traitement ou projection :** On peut envisager de « forcer » la solution numérique à rester positive en projetant les valeurs négatives sur 0 (par exemple, en remplaçant x[i]x[i] par max⁡(x[i],0)\max(x[i], 0) à chaque étape). Cependant, cette approche est souvent une astuce numérique qui masque potentiellement un problème de modélisation ou de paramétrage.

**Recommandations pratiques :**

* **Affiner la discrétisation** : diminuer Δt\Delta t en augmentant NN pour améliorer la précision de la somme approchée de l’intégrale.
* **Vérifier et calibrer les paramètres** : assurez-vous que
  + rlogr\_{\text{log}} et KK sont choisis de sorte que la croissance naturelle compense, au moins partiellement, l’extraction u(t)x(t)u(t)x(t) au début.
  + Le paramètre pp (dans le terme de rétention) est réaliste et ne décale pas trop l’effet du passé.
* **Contraintes explicites sur** xx : si possible, ajouter une contrainte x(t)≥0x(t)\ge0 dans le solveur (par exemple en utilisant une formulation souple ou une pénalité très importante lorsque x(t)x(t) est négatif).
* **Choix du contrôle initial** : Testez si une initialisation de uu moins agressive (par exemple une valeur initiale inférieure à E/2E/2) permet de mieux préserver la positivité de x(t)x(t).

En résumé, vous obtenez des valeurs négatives probablement parce que (i) la méthode d’intégration simple introduit une erreur dans la résolution du système de type Volterra, et (ii) la configuration des paramètres (notamment un effort trop élevé par rapport au potentiel de croissance) engendre une dynamique qui n’est pas économiquement ou biologiquement réaliste. Essayez de revoir ces aspects afin d’obtenir une solution conforme aux attentes physiques.

N’hésitez pas à explorer ces modifications et à affiner la modélisation pour mieux correspondre à la réalité biologique.

Avec toutes les simulations rélisées, la biomasse décroît systémtiquement bien que sa valeur tend vers une limite très faible (~2 unités). Quels paramètres influent sur le fait que la biomasse croît ou décroît mais lentement ?

Plusieurs paramètres entrent en jeu dans votre modèle et influencent la dynamique de la biomasse. Le fait que la biomasse décroisse lentement (et tende vers environ 2 unités) s’explique souvent par un équilibre entre la croissance naturelle, la dissipation (ce qui atténue l’effet du passé) et le prélèvement (la récolte). Voici les principaux paramètres et leur effet :

1. **Le taux de croissance logistique** rlogr\_{\text{log}} **:** Dans le modèle, la croissance naturelle est décrite par la fonction

F(x)=rlog x(1−xK).F(x) = r\_{\text{log}}\, x\Bigl(1-\frac{x}{K}\Bigr).

Un rlogr\_{\text{log}} élevé tend à accélérer la croissance de la population lorsque xx est faible. Inversement, si rlogr\_{\text{log}} est faible, même en l’absence d’extraction la croissance sera limitée et l’équilibre atteint pourra être très bas.

1. **La capacité de saturation** KK **:** C’est la borne supérieure théorique de la biomasse lorsqu’aucune extraction n’est effectuée. Un KK plus grand offre un potentiel de croissance plus important, tandis qu’un KK faible limite l’état maximal possible. Même si la croissance suit une loi logistique, l’extraction peut forcer l’état vers une valeur bien inférieure à KK.
2. **Le contrôle ou effort d’extraction** u(t)u(t) **:** L’effet de l’extraction se traduit par le terme négatif −u(t) x(t)-u(t) \, x(t). Si u(t)u(t) est de manière continue trop élevé (même s’il reste dans les bornes admises), il pourra « trop prélever » par rapport à la capacité de renouvellement de la biomasse, faisant décroître l’état vers un équilibre bas. Même un effort modéré mais constant peut conduire à un équilibre faible si la croissance naturelle ne compense que partiellement le prélèvement.
3. **Le paramètre de dissipation** pp **dans la fonction de rétention** r(t−s)=exp⁡(−p (t−s))r(t-s)=\exp\bigl(-p\,(t-s)\bigr) **:** Ce paramètre détermine le poids des contributions du passé dans l’intégrale. Un pp élevé fait que l’effet des contributions anciennes s’atténue rapidement, ce qui diminue la contribution cumulée à la croissance de x(t)x(t). Ainsi, même si la fonction logistique pourrait favoriser une croissance plus forte, un pp important atténue cet effet et limite la biomasse finale.
4. **La discrétisation (pas de temps** Δt\Delta t**) et le schéma numérique :** Enfin, le choix de la méthode d’intégration (ici, la méthode du rectangle) et la taille de Δt\Delta t influencent la précision de la simulation. Un Δt\Delta t trop grand peut introduire des erreurs numériques qui, dans certains cas, amplifient la décroissance ou induisent des erreurs (bien que le phénomène que vous décrivez – une décroissance lente vers un très faible équilibre – puisse correspondre à un équilibre théorique).

En résumé, si dans vos simulations la biomasse décroît progressivement pour tendre vers environ 2 unités, cela suggère que, compte tenu des paramètres choisis, la croissance naturelle (déterminée par rlogr\_{\text{log}} et modulée par KK) ne compense qu’à peine l’impact de l’extraction u(t)u(t) et/ou de l’atténuation due à pp. Pour obtenir une croissance (ou un équilibre plus élevé) ou à l’inverse une décroissance plus rapide, vous pourriez expérimenter avec :

* **Augmenter** rlogr\_{\text{log}}**:** Pour renforcer la croissance intrinsèque.
* **Augmenter** KK**:** Pour autoriser un potentiel de biomasse plus important.
* **Diminuer** u(t)u(t) **(ou adopter un profil d’extraction plus modéré):** Pour réduire l’effet négatif sur la biomasse.
* **Diminuer** pp**:** Ce qui renforce l’influence des contributions passées, améliorant ainsi la croissance cumulée.

En ajustant ces paramètres, vous pourrez influer sur la dynamique de la biomasse, obtenir une croissance plus rapide ou un équilibre plus élevé, ou alors renforcer encore l'effet d’extraction si tel est votre objectif. N’hésitez pas à expérimenter et à observer comment chaque paramètre agit sur l’évolution de x(t)x(t).

Comment optimiser le paramètre d'extraction pour améliorer la biomasse?

Pour améliorer la biomasse tout en récoltant, il faut « optimiser » le contrôle d’extraction u(t)u(t) afin d’équilibrer le prélèvement et la croissance naturelle. Autrement dit, il s’agit de choisir une stratégie d’extraction qui contribue au profit tout en préservant ou en améliorant la taille de la population.

Voici quelques pistes et approches pour optimiser ce paramètre :

**1. Re-formulation en problème de contrôle optimal**

Le modèle que vous avez est (dans sa version continue) :

x(t)=x0+∫0t[e−p(t−s) F(x(s))−u(s) x(s)]ds,x(t) = x\_0 + \int\_0^t \Bigl[ e^{-p(t-s)}\, F(x(s)) - u(s)\, x(s) \Bigr] ds,

avec

F(x)=rlog x(1−xK),F(x) = r\_{\text{log}}\, x\left(1 - \frac{x}{K}\right),

et l'objectif (profit actualisé) est

J(u)=∫0Te−δt u(t) x(t) dt.J(u) = \int\_0^T e^{-\delta t} \, u(t)\, x(t)\, dt.

Pour améliorer la biomasse, il peut être judicieux de ne pas maximiser uniquement le profit issu de la récolte, mais de prendre en compte la **durabilité** : par exemple, ajouter dans l’objectif un terme de pénalité lorsque x(t)x(t) est trop faible ou imposer une **contrainte d’état** (par exemple x(t)≥xmin⁡x(t) \ge x\_{\min}).

**2. Principe du maximum de Pontryagin**

Pour résoudre le problème optimal, on peut appliquer le principe du maximum de Pontryagin. On définit une fonction Hamiltonienne

H(t,x,u,λ)=e−δt u x+λ(t) [e−p(t−s) F(x)−u x]H(t, x, u, \lambda) = e^{-\delta t}\, u\,x + \lambda(t)\,\Bigl[e^{-p(t-s)}\, F(x) - u\, x\Bigr]

(pour une rédaction rigoureuse, il faut faire attention au fait que la dynamique est de type Volterra – ce qui peut demander de travailler dans un cadre légèrement généralisé ; on peut toutefois, dans une première approche numérique, travailler avec la version discrétisée).

Les conditions nécessaires indiquent notamment que, pour presque tout tt, le contrôle optimal u∗(t)u^\*(t) maximise HH parmi les valeurs admissibles (généralement 0≤u(t)≤E0 \le u(t) \le E). Cette condition va généralement conduire à une loi de commande « bang–bang » ou « bang–singular » si l’on rencontre des zones où la fonction coût est plate. Pour préserver la biomasse, il est souvent optimal de réduire u(t)u(t) lorsque x(t)x(t) est faible (pour éviter un prélèvement trop important) et de relever u(t)u(t) lorsque la biomasse est abondante.

**3. Ajustement de la formulation pour « améliorer » la biomasse**

**a) Stratégie Multi‑objectif**

Vous pouvez reformuler l’objectif par une fonction coût composée qui combine, par exemple, le profit et une pénalité associée à un niveau de biomasse trop faible. Par exemple, vous pourriez choisir de maximiser :

J(u)=∫0T(e−δt u(t) x(t)−γ  P(x(t)))dt,J(u) = \int\_0^T \left( e^{-\delta t}\, u(t)\, x(t) - \gamma\; P(x(t)) \right) dt,

où P(x)P(x) est une pénalité (par exemple, P(x)=max⁡{0,xmin⁡−x}P(x) = \max\{0, x\_{\min} - x\}) et γ>0\gamma > 0 est un coefficient qui pèse l’importance de conserver la biomasse.

**b) Choix d’une stratégie adaptative**

Une bonne stratégie « optimisée » consiste à extraire moins lorsque la biomasse se rapproche d’un niveau critique et à augmenter l’extraction lorsqu’elle est abondante. Cette stratégie adaptée peut, par exemple, se traduire par :

* u(t)u(t) **faible** quand x(t)x(t) est proche d’un minimum acceptable;
* u(t)u(t) **plus élevé** quand x(t)x(t) dépasse un certain seuil.

En d’autres termes, vous pouvez paramétrer u(t)u(t) de manière à ce que l’effort d’extraction soit une fonction décroissante de 1x(t)\frac{1}{x(t)} (ou d’un indicateur de stress sur la population) dans certaines plages, et ainsi « préserver » la biomasse.

**4. Approche numérique (en Python) pour optimiser u(t)u(t)**

Dans le code précédent, la fonction objectif était le profit pur. Pour essayer d’optimiser le paramètre de récolte pour « améliorer » la biomasse, vous pouvez procéder de plusieurs façons :

* **Améliorer la dynamique** : Par exemple, en ajoutant une pénalité pour les faibles biomasses dans l’objectif. Vous pourriez modifier la fonction objective(u, params) pour inclure un terme du type

python

penalty = gamma \* sum(max(0, x\_i\_min - x\_i)\*\*2 for x\_i in x)

return -profit + penalty

avec x\_i\_{\min} le seuil minimal souhaité et γ\gamma un coefficient de pénalité.

* **Utiliser des contraintes supplémentaires** : En ajoutant une contrainte discrète (via un solveur qui accepte des contraintes non linéaires) pour forcer xi≥0x\_i \ge 0 ou xi≥xmin⁡x\_i \ge x\_{\min}.
* **Paramétrer le contrôle** : Vous pouvez paramétrer uu par une fonction (par exemple, une fonction affine ou sigmoïdale dépendant de xx) et optimiser les coefficients de cette fonction afin d’obtenir la meilleure trajectoire de x(t)x(t).

Par exemple, vous pourriez considérer une loi de commande du type :

u(t)=umax⁡⋅σ(α(x(t)−xcrit))u(t) = u\_{\max} \cdot \sigma\Bigl(\alpha (x(t)-x\_{\mathrm{crit}})\Bigr)

où σ(z)=11+e−z\sigma(z)= \frac{1}{1+e^{-z}} est la sigmoïde, xcritx\_{\mathrm{crit}} un seuil critique, et α\alpha un paramètre de pente. Ensuite, vous n’optimiserez plus une séquence uiu\_i, mais les paramètres α\alpha et xcritx\_{\mathrm{crit}}.

**5. Conclusion**

En résumé, pour optimiser le paramètre d'extraction et améliorer la biomasse :

* **Revoyez les coefficients de croissance** (rlogr\_{\text{log}} et KK) et le paramètre de dissipation pp pour vous assurer que la croissance potentielle est suffisamment forte par rapport à l’extraction.
* **Ajustez la stratégie d’extraction** en diminuant u(t)u(t) quand x(t)x(t) devient trop faible (via une contrainte ou un terme de pénalité dans l’objectif) afin d’éviter une décroissance excessive.
* **Utilisez une formulation de contrôle optimal** (avec, par exemple, le principe du maximum de Pontryagin) ou une approche par optimisation directe (en discretisant la dynamique) pour rechercher la loi u∗(t)u^\*(t) optimale.
* **Expérimentez avec des contrôles paramétrés** pour imposer une dépendance adaptative de u(t)u(t) en fonction de x(t)x(t).

En appliquant ces stratégies, vous pourrez « optimiser le paramètre d’extraction » de façon à soutenir une biomasse plus élevée ou à ralentir sa décroissance dès les premières années.

Si vous avez besoin d’un exemple de mise en œuvre avec une formulation paramétrée (par exemple avec la sigmoïde) ou d’autres détails sur l’optimisation numérique, je peux vous proposer un code complémentaire.

# Paramètres du modèle et valeurs usuelles

|  |  |  |
| --- | --- | --- |
| Symbole | Signification | Valeurs typiques / Usuelles |
| r*r* | Taux de croissance intrinsèque de la biomasse | 0.2 à 1.0 (souvent 0.5 à 0.8) |
| K*K* | Capacité de charge de l’écosystème | 500 à 2000 (souvent 1000) |
| x0*x*0 | Biomasse initiale | 0.2 K à 0.8 K (ex : 500 si K=1000*K*=1000) |
| γ*γ* | Coefficient de dissipation cumulative | 0.005 à 0.05 (souvent 0.01) |
| T*T* | Horizon temporel (années) | 5 à 20 (souvent 10) |
| p*p* | Prix unitaire de la biomasse récoltée | 5 à 20 (souvent 10) |
| c*c* | Coût unitaire de l’effort de récolte | 1 à 5 (souvent 2) |
| δ*δ* | Taux d’actualisation | 0.03 à 0.1 (souvent 0.05) |
| umax*umax* | Effort maximal admissible | 0.1 K à 0.3 K (ex : 200 si K=1000*K*=1000) |

**Références et exemples**

* **Yezza (1991), pages 162-163** :
  + r=0.75
  + K=1000
  + x0=500
  + γ=0.01
  + T=10
  + p=10
  + c=2
  + δ=0.05
  + umax=200
* **Littérature classique (voir Clark, Mathematical Bioeconomics)** :
  + Les valeurs de r*r* et K*K* sont souvent choisies pour représenter une population de poissons, d’arbres ou d’autres ressources renouvelables, avec r*r* dans [0.2, 1.0] et K*K* dans [2000].

**Remarques**

* **Le choix de γ*γ*** dépend du phénomène de dissipation ou de mortalité cumulative que l’on souhaite modéliser. Une valeur plus élevée accélère la décroissance de la biomasse.
* **Le taux d’actualisation δ*δ*** reflète la préférence pour le profit immédiat par rapport au profit futur.
* **Le coût unitaire c*c*** peut être ajusté pour représenter des techniques de récolte plus ou moins coûteuses.
* **L’effort maximal umax*umax*** est souvent fixé pour garantir que la biomasse ne soit pas surexploitée.

**Extrait de la thèse (p. 162-163, adapté)**

*« Pour les simulations numériques, nous avons utilisé les valeurs suivantes : r=0.75r=0.75, K=1000K=1000, x0=500x0=500, γ=0.01γ=0.01, T=10T=10, p=10p=10, c=2c=2, δ=0.05δ=0.05, umax=200umax=200. »*

**En résumé :**  
Les valeurs ci-dessus sont standards pour ce type de modèle. Elles peuvent être adaptées selon l’espèce, l’écosystème ou la problématique économique étudiée. Pour toute simulation ou optimisation, il est recommandé de réaliser une analyse de sensibilité autour de ces valeurs.