Bou Orm Khodor
Delamotte Hippolyte
Mateos Estaban
Rappaport Thomas

MIPI21

ARE 2019 :

Dynamique des populations et modèle proies/prédateurs

Dynamique des populations et modèle proies/prédateurs

SOMMAIRE

Résumé

Pour modéliser l'évolution des populations des espèces d'un écosystème, nous avons décidé d'adopter une approche comparative de deux modèles. Le premier modèle se base sur un couple d'équations différentielles (établies par Lotka et Volterra), décrivant l'évolution des populations dans une simple chaîne à deux espèces. L'enjeu était d'adapter ces équations à un modèle discret. Le deuxième modèle étudie le comportement de chaque individu selon son état (faim, âge, sexe...). Il s'agissait ici d'étudier un modèle fonctionnant selon des « lois naturelles ». On a choisi d'utiliser des animations et des graphiques comme moyen de présentation des résultats. L'étude des résultats se fera selon deux perspectives. La première sera une étude de l'importance des paramètres à l'intérieur du modèle. La deuxième sera une comparaison inter-modèles pour comparer les deux modèles et savoir s'ils correspondent et leurs limites. Nous pouvons ainsi trouver une stabilité en cas d'écosystème à deux espèces, mais pas lorsqu'il y en a plus.

Introduction

La création de ce modèle de dynamique de population nous permet de nous pencher sur plusieurs questions : Quels paramètres sont les plus impactant quant à l'évolution d'une population ? De simples équations sont elles suffisantes pour décrire la réalité? Les comportements individuels aboutissent-ils à des comportements généraux de groupe (*cf, Game of life)* Les paramètres généraux d'âge, de faim et de sexe sont ils trop réducteurs ou sont ils suffisants pour un modèle réaliste ?
Pour répondre à ces questions nous avons décidé de nous mettre à 4, Bou Orm Khodor, Delamotte Hippolyte, Mateos Esteban et Rappaport Thomas; et envisagé des méthodes de résolution aux problèmes. Nous avons pensé à combiner les modèles ensemble pour voir si cela enrichissait le modèle mais ceci s'est avéré trop coûteux en terme de code et de temps et nos quelques tentatives n'étaient pas très prometteuses, nous avons aussi pensé à utiliser des photos pour représenter les individus sur la carte mais ça devenait incompréhensible pour des populations trop grandes et donc de petites photos. Finalement nous avons décidé de séparer les modèles et les comparer. Une représentation graphique s'est imposé pour bien représenter l'évolution de la population; nous avons choisi des courbes des populations en fonction du temps car nous les estimons les plus parlantes. Nous avons aussi décidé décidé de représenter l'évolution de la population dans un espace(chaque espèce représentée par des points colorés). Nous avons codé en python pour donner naissance a cette simulation (avec l'aide de numpy, matplotlib, pygame et tkinter). Dans ce rapport nous allons dans un premier temps présenter le paramètres initiaux, pourquoi nous avons choisis ces paramètres pour effectuer la simulation et comment chaque paramètre affecte la population. Ensuite nous allons nous focaliser sur une explication du code et comment nous avons procédé pour l'écrire en expliquant le fonctionnement de chaque fonction. Finalement nous allons nous focaliser sur une analyse des résultats; de chaque modèle pour ensuite les comparer. On présentera des captures d'écran des graphes et des liens vers des vidéos des cartes.

Présentation de la thématique

Notre travail repose sur un système multi agent. Nous avons décidé de réunir ces agents avec l'environnement dans une même matrice, ce qui permet de faciliter les relations agents-agents et agents-milieu. Premièrement, le système sera défini par 2 espèces différentes en tant qu'agents, d'une modélisation d'un terrain avec des zones de plantes comme environnement et un ensemble de règles définissant les interactions des uns avec les autres. Dans un second temps, un autre agent représenté par un prédateur encore plus puissant sera introduit pour complexifier les interactions et tenter de se rapprocher le plus possible de la réalité.

Initialement, nous avons une matrice population qui contient uniquement les animaux (Annexe 1), et une matrice terrain contenant la végétation (Annexe 1). La population initiale et le taux de végétation est décidé initialement, avant de réunir les deux matrices pour enfin lancer la simulation (Annexe 1). Lors de la simulation, les animaux pourront se déplacer librement sur la matrice et interagir entre-eux et avec le milieu

pour la reproduction et la nutrition. Un système de coordonnées est utilisé pour appeler les animaux indépendamment.
Chaque agent est défini par : son sexe, son âge (en mois), sa jauge de faim et sa durée de gestation. Chaque mois, il perd de l'énergie qu'il récupère ensuite en mangeant. Chaque espèce peut ainsi mourir de vieillesse, de faim ou alors en se faisant manger par un prédateur.

Les agents se déplacent en priorité vers des cases ayant de la nourriture s'il ont faim. Dans le cas contraire, s'ils ont un age suffisamment élevé et ne sont pas en période de gestation, ils vont se déplacer vers des individus de même espèce et de sexe opposé dans le but de perpétuer son espèce. Si l'agent se retrouve sur une case adjacente à sa cible, il peut alors se nourrir ou se reproduire. Lors de la reproduction, l'enfant naît directement par soucis de simplification mais la femelle entre en période de gestation. Ci-contre voici une représentation des matrices Terrain et Animaux, et de leur réunion.
Dans la matrice animale, chaque couleur représente un agent. Les 2 nuances de chaque couleurs permettent de mieux représenter les deux sexes dans chaque espèce. La matrice Terrain est composé de buissons, la probabilité de présence de ces buissons dépend de l'environnement sélectionné initialement.
Lorsque que le programme est exécuté, les résultats sont présentés sous forme d'animations pour la matrice, permettant aisément de constater l'évolution au cours du temps du milieu et de la population. Également, toutes les statistiques sont accessibles par l'intermédiaire de courbes pour rendre comptes des données démographiques et environnementales. Chaque espèce est dans un dictionnaire. Dans ce dictionnaire, la clé est la position de chaque animal, et les valeurs représentent différentes propriétés de l'animal.

Description du code

A)Les paramètres

Nous allons ici faire une liste exhaustive des paramètres globaux ainsi qu'une description rapide de leur utilité. On peut dans un premier temps gérer la taille de la matrice dans laquelle évoluera la population
avec le paramètre Dimension qui correspond au nombre de lignes et de colonnes (notre Terrain est carré). Il est également possible de gérer les ressources initiales de végétation à l'aide du paramètre Environnement. Lorsque la probabilité est à 100% toutes les cases seront recouvertes de ressources, si la probabilité est à 0 % il n'y aura aucune ressource sur les cases.
A la création de la matrice Animaux, on peut gérer le nombre d'animaux initialement avec le paramètre List_number_species : Le 1er élément correspond à l'espèce la plus faible, le 2ème élément à l'espèce intermédiaire et le dernier élément à l'espèce la plus forte.
Plusieurs paramètres permettent de modifier les indices vitaux des animaux : La gestation de chaque espèce avec les paramètres « gestation », la jauge de faim perdue à l'aide du paramètre « faim perdue » qui permet de prendre en compte la différence de chaque espèce, en effet chaque espèces n'ont pas faim au même moment. Le paramètre « faim » permet de savoir à partir de quel stade les agents préféreront manger que faire tout autre action. Enfin on définit un âge de maturité sexuelle pour chacune des espèces avec le paramètre « amr ».

B) Le code

Le principe du code est relativement simple, son exécution repose sur la fonction "cgmt()" qui appellera toutes les autres fonction. Cette fonction parcourt les dictionnaires de chaque espèces, et applique différentes fonction selon chaque animal : il y a tout d'abord une vérification d'âge et de faim (s'il est trop âgé où qu'il n'a plus d'énergie, il décède) ensuite, s'ils sont en dessous du seuil de faim, ils parcourent le Terrain à l'aide de la fonction périmètre et si la cible est situé dans une case adjacente, alors l'animal la fonction prédation est appliquée en cas d'interaction entre le prédateur et sa proie. Sinon il s'agit de la fonction prédation_plante qui est appliquée. Si la jauge de faim est suffisamment remplie, la fonction périmètre est appliquée cette fois pour trouver un animal de la même espèce mais de sexe opposée à condition d'avoir atteint la maturité sexuelle. Enfin, si aucune des conditions n'est appliquée, alors l'animal va se déplacer de

manière aléatoire. Si lorsqu'un animal trouve une cible mais qu'elle n'est pas à côté, celui-ci se déplace dans la matrice à l'aide de la fonction déplacement.
Vous pouvez trouver un schéma explicatif de cette fonction dans l'annexe 2.
Nous allons maintenant détailler le fonctionnement de certaines fonctions complexes pour comprendre la suite il faut connaître quelques principes globaux à propos de notre code. Le premier est que nos agents sont contenu dans différents dictionnaires, ils ont pour clé leur position et leur contenu est une liste contenant dans l'ordre l'age des agents, la jauge de faim et en troisième élément leur période de gestation (cet élément est présent dans chaque dictionnaire mais est utile seulement chez les femelles).

La fonction déplacement :
La fonction déplacement utilise des cibles afin des savoir quel chemin va prendre l'agent. Ces cibles sont définies par la fonction 'perimetre_vision' qui parcourt le Terrain autour de l'animal avec un certain périmètre et qui localise la cible la plus proche. En prenant en compte cette cible, la fonction de déplacement créera une règle de préférence pour que le prédateur atteigne sa cible le plus rapidement possible sachant que pour chacune des cases du Terrain il ne peut y avoir qu'un seul animal.

La fonction affichage :
Cette fonction permet de transformer chaque case de la matrice Terrain en valeur rgb pour ensuite l'afficher. Ainsi les plantes, représentées par la valeur 0, sont transformées en vert, le sol, représenté par la valeur 1 est transformé en jaune, la proie la plus faible, (de valeur 2 ou -2 en fonction du sexe) est transformée en marron clair ou foncée, le prédateur (de valeur 3 ou -3) est violet, et enfin, le prédateur qu'on ajoute pour complexifier le modèle est transformé en bleu.

Analyse des résultats :

Nous allons dans cette partie étudier l'évolution des espèces en fonction de différents paramètres afin de mieux comprendre les raisons de leur croissance et également de comprendre comment atteindre les résultats obtenus par les équations de Lotka-Volterra. Pour cela nous allons considérer une matrice de 36 x 36 cases tout au long de nos simulations afin que nos résultats ne soient pas impactés par la taille de l'environnement.

I/ Evolution des proies seules

Pour commencer nous allons étudier l'évolution des proies en globalité sur toutes la carte :

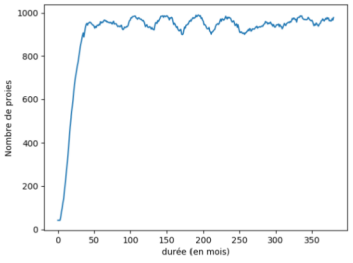
A) Mode d'évolution des proies :
On constate que dans les premiers jours ils ne cessent de se multiplier jusqu'à atteindre une valeur limite ou leur nombre devient stable (oscillant autour d'une valeur moyenne). On peut expliquer cela par le fait que les proies n'ont aucun prédateur sur la carte, ils vont donc se multiplier jusqu'à occuper tout l'espace, les oscillations étant dues au manque de place qui entraîne une légère chute de la population.

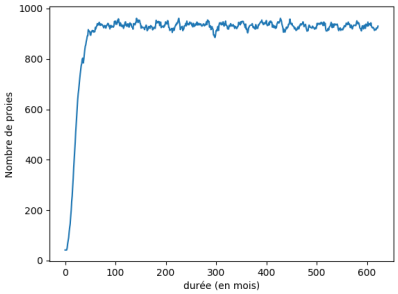
B) Sensibilité aux paramètres :

1/ Perte d'énergie

Nous allons ici étudier l'impact du paramètre de perte d'énergie à chaque mois, sans modifier la jauge à laquelle la proie va préférer se nourrir à se reproduire :

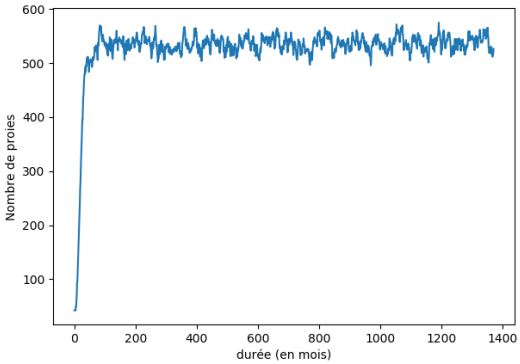
Dans cette première simulation, nous allons considérer que les espèces ne perdent jamais d'énergie, ainsi ils essaieront toujours de se reproduire. On peut très facilement apercevoir le phénomène de générations, en effet les prédateurs se reproduisent tous dans la même période, et attendent ensuite la fin de la gestation des femelles avant de se reproduire à nouveau. Ainsi, étant donné qu'ils naissent tous en même temps, ils décèdent aussi en même temps, ce qui donne de forte oscillations. Ce phénomène se stabilise au cours du temps, puisque les générations se mélangent.





Dans cette deuxième simulation la jauge d'énergie de nos proies diminuaient de 10, on observe une stagnation autour des 1000 proies, avec oscillations. Ce phénomène est trouvable entre 5 et 15 environ, sachant que plus cette valeur augmente, plus la stagnation est basse (environ 950 lorsque le paramètre est à 15).

Dans cette dernière simulation, la jauge d'énergie de nos proies diminuaient de 20 à chaque itération. La stagnation est d'environ 550 proies. A cette valeur, on peut observer que les proies ne peuvent plus s'aventurer dans des zones sans nourriture, c'est pourquoi on peut voir des zones totalement vides. En effet, dès qu'un animal s'écarte de sa nourriture, il est obligé de s'y rapprocher rapidement pour se nourrir. De plus, les oscillations de l'évolution démographique sont beaucoup plus importantes, celles-ci sont explicables par le fait que la végétation est beaucoup plus ciblées, il y a ainsi beaucoup plus de proies souhaitant se nourrir que de plante pour les nourrir, elles meurent donc de faim.



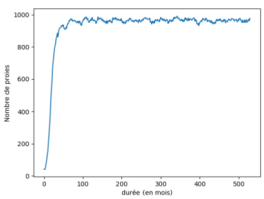
Ce dernier cas permet donc de nous questionner sur le paramètre de l'environnement, si le taux de végétation joue un rôle important dans l'évolution des proies.

2/ Taux d'apparition de la végétation

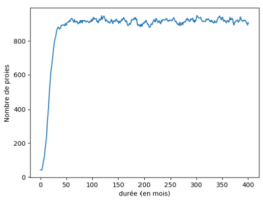
Dans cette simulation, nous allons rester avec le taux d'énergie perdue à 20, et avec 42 proies initialement :

Comme on peut le voir avec ces courbes, plus la probabilité des végétations est élevé, plus le nombre de proies tend vers une valeur élevée. De plus, on peut apercevoir que la probabilité de la végétation joue un rôle important dans les oscillations de l'évolution démographique : comme expliqué précédemment, les oscillations étaient causées d'une part par un manque de place sur le terrain. Ici, on peut aussi apercevoir que les oscillations sont aussi et principalement causées par le rapport entre le nombre d'animaux ayant faim et le nombre de nourriture proposée : si ce rapport est inférieur à 1, c'est à dire qu'il y a plus d'animaux ayant faim que de végétation disponible, alors les oscillations seront très importantes. Cependant, si ce rapport est supérieur à 1, les oscillations seront seulement causées par le manque d'espace et seront donc faibles. De plus, moins il y a de nourriture disponible, plus on peut voir les animaux se réunir uniquement autour des plantes, laissant donc des espaces vides.

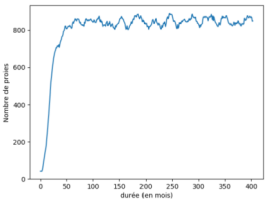
1/2 :



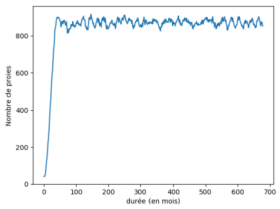
1/4 :

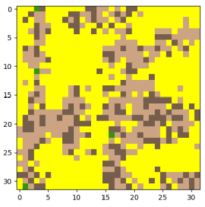


1/8 :



1/16 :





Voici le terrain après 400 mois, comme expliqué précédemment, on peut voir un regroupement des espèces autour des végétations, et des zones vides.

II-Cohabitation proies/prédateurs

Dans cette simulation, nous allons étudier la cohabitation entre deux espèces, une proie et un prédateur. En étudiant la réaction en fonction des paramètres, nous avons trouvé l'ensemble des résultats prédit dans la résolution des équations le Lotka-Volterra : (Chacun des paramètres sélectionnés sont trouvables en Annexe)

Cas 1 :

Tout d'abord, il y a le cas où les prédateurs n'ont pas le temps de se développer suffisamment, ainsi ceux-ci décèdent et les proies se développent comme nous avons étudié précédemment. Les paramètres choisis sont en Annexe.

Ce cas est ainsi pas très intéressant dans l'objectif de la survie de l'écosystème.

Cas 2 :

Dans ce cas, les prédateur ne laissent pas le temps aux proies de se développer. Ensuite, étant donné qu'il n'y a plus de proies, les prédateurs ne peuvent plus se nourrir.Ils décèdent donc de faim, l'écosystème est mort. Il est tout de même intéressant de voir que malgré l'absence de nourriture, les prédateurs tentent de survivre à travers la reproduction, bien que cette solution ne fonctionne pas.

Cas 3 :

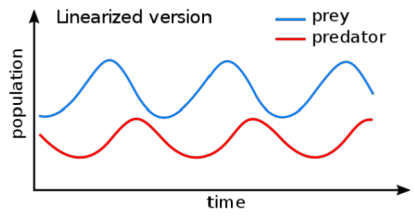
Dans ce cas, les proies et les prédateurs vivent en quasi-équilibre, en effet l'équilibre dépend des positions initiales, il y a ainsi encore 3 cas repérables : (Pour voir les vidéos, regarder l'annexe 3)

1er cas : les prédateurs et les proies vivent d'abord une période d'équilibre, sauf qu'à un moment, les prédateurs empêchent les proies de s'enfuir, en effet celles-ci n'ont plus d'espace pour se déplacer où se reproduire, ainsi les proies meurent chassées. Très rapidement après, étant donné que les prédateurs n'ont plus de cible, ceux-ci meurent de faim (courbe en Annexe 4).

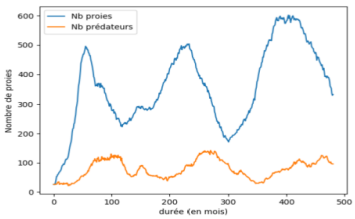
2 ème cas : Lorsque les proies s'enfuient, elle ne sont poursuivies seulement par des prédateurs mâles. Ainsi, mêmes s'ils sont rassasiés, il ne peuvent pas se reproduire et meurent de vieillesse.(courbe en Annexe 4)

3 ème cas : Comme on peut le voir dans cette vidéo, il y a un jeu de course poursuite entre les prédateurs et les proies. En effet, à chaque fois, les prédateurs attaquent leurs proies, cependant il y en a quelques unes qui réussissent à s'échapper et à se redévelopper. Ce petit groupe est toujours suivi par un petit groupe de prédateur, qui va donc attaquer les proies jusqu'à ce qu'elles s'enfuient et ainsi de suite. Évidemment on peut imaginer qu'au bout d'un moment les autres cas peuvent apparaître, mais ceux-ci sont uniquement causés par nos conditions initiales.

Voici la courbe qui était attendu d'après les équations de Lotka- Volterra :



Et voici la courbe obtenue avec ce 3ème cas :



Malgré le fait qu'on trouve que les deux courbes soient différentes au premier regard, on peut tout de même apercevoir de nombreuses similitudes qui permettent de dire que les deux sont liées : tout d'abord, dans les courbes des proies, si on retire les mini-oscillations de la courbe expérimentale (qui sont causées par la faim , l'âge et le terrain limité), on peut voir que les deux courbes ont la même évolution sinusoïdale, la courbe expérimentale est juste en croissance car il s'agissait du début de l'expérimentation, et on peut donc imaginer qu'une fois qu'elle a atteint sa valeur max, elle aurait une évolution sinusoïdale comme la courbe des équations. De plus, les courbes de prédateurs ont des similitudes aussi, en effet il y a un écart (un pas) dans le temps entre le max des proies et des prédateurs,

de plus, les deux courbes ont des évolutions périodiques avec des amplitudes moins élevées et les amplitudes des proies. Enfin, la valeur de la population est toujours inférieure à celle des proies.
On peut donc dire que cette solution des équations de Lotka-Volterra représente bien un écosystème réduit, cependant, étant donné l'allure des courbes, on peut se douter que cette allure ne va pas durer indéfiniment. Ce problème est causé par notre modèle et nos conditions, telles que le fait que les deux espèces se déplacent à une vitesse identique, qu'il y a une limite de terrain et que le prédateur trouve toujours sa proie et réussit toujours à la manger (en vrai, on peut imaginer des proies se cachant dans un terrier par exemple).

Ainsi, en trouvant de bons paramètres qui sont plutôt réalistes, il est possible d'avoir un écosystème qui dure dans notre programme.

III- Ajout d'un prédateur 'ultime'
Pour se rapprocher encore plus de la réalité, nous avons essayer d'ajouter une troisième espèce, celle-ci serait un prédateur qui se nourrirait des deux autres espèces sans distinction.
Nous savons que le prédateur régule la population des proies, puisque sans prédateur les proies prolifèrent librement, il s'agit d'ailleurs des résultats vus précédemment, mais cela se complexifie lorsqu'on ajoute une troisième espèce : en fonction des interactions entre chaque espèces, on peut très bien imaginer un écosystème où le prédateur ultime mange le prédateur intermédiaire qui lui-même se nourrit des proies. Cependant, on peut aussi imaginer un écosystème où le prédateur ultime mange les deux autres espèces.

En prenant en compte ces deux cas de figures on se doute bien que l'écosystème n'évoluera pas de la même manière, dans la 1ère option, on a une hiérarchie qui est plus avantageuse pour les proies car la population de son prédateur est régulée par un autre prédateur. En revanche dans la deuxième option la survie de l'espèce proie est plus difficile car elle aura deux prédateurs.

Il nous a semblé que dans la nature la deuxième option était plus abondante, on a donc décidé de choisir ce modèle. Dans ce modèle, nous avons ainsi fait varier différents paramètres afin de trouver un équilibre.
Comme nous l'avons expliqué un peu plus tôt le prédateur régule la population de sa/ses proies. Ainsi, plus la population des prédateurs sera grande plus la « régulation » sera forte et moins les espèces chassés auront de chance d'être pérenne. On peut donc déduire que la population initiale aura une très fort impacte quant a la survie de l'écosystème, plus la population initiale des prédateurs sera grande plus la chute et l'extinction de l'écosystème sera rapide. Il s'agit donc du paramètre le plus influent pour l'écosystème.
Nous avons ainsi testé de modifier de nombreux paramètres dans la limite du réel (une espèce ne peut pas se reproduire le jour de sa naissance par exemple), et à chaque fois, nous n'avions pas d'équilibre, au contraire le prédateur ultime ressortait presque tout le temps vainqueur, ou alors il décédait rapidement et on retournait à un modèle à deux espèces. Ces résultats peuvent s'expliquer avec plusieurs facteurs :

- le codage du terrain fait qu'il ne peut y avoir qu'un seul animal par case, ce qui en plus de ne pas être réaliste (notre modèle est une simplification de la réalité, c'est donc normal d'avoir quelques actions peu réalistes), pose surtout le problème du cas ou l'espèce intermédiaire va être bloquée et ne plus pouvoir ni se déplacer ni se reproduire, car elle sera entouré de l'espèce proie, de l'espèce ultime ou des deux, provoquant ainsi la mort par « asphyxie » du groupe piégé, on peut d'ailleurs remarquer que c'est un problème que l'espèce ultime ne peut pas rencontrer car elle elle peut manger tout obstacle et ainsi se frayer un chemin dans toute situation, il s'agit ainsi d'une raison pour expliquer pourquoi elle ressort toujours vainqueur.

- Notre terrain est limité, en effet notre modèle fait évoluer l'écosystème dans une « cage », dans un espace limité, ainsi les proies ne peuvent pas, premièrement s'enfuir (ou dans la limite du possible), deuxièmement elles ne peut pas se reproduire de manière suffisante pour que la reproduction surpasse le taux de prédation. Tout cela est confirmé par les tests : en effet, avec des certains paramètres permettant une survie instable, lorsque l'on à doublé la taille du terrain, on est passé d'un écosystème qui mourrait en 200 mois (soit à peu près 16 ans ce qui est déjà pas mal), à un écosystème qui meurt en 300 mois (soit à peu près 25 ans). Ainsi en doublant la taille de la carte, on multiplié « l'espérance de vie » de notre écosystème par 1,5. On peut donc supposer qu'il s'agit d'une évolution logarithmique, et donc que si on étendait la carte à l'infini on pourrait obtenir un écosystème stable (Annexe 4).

Conclusion :

En somme, grâce à notre programme, nous sommes capable d'approcher l'évolution d'une espèce, avec ou sans prédateur. En jouant sur la sensibilité des paramètres on observe que le modèle qui a été proposé par Lotka et Volterra est réaliste sur deux espèces. En effet, l'espérance de vie des agents, la densité en ressources et la faim des agents jouent un rôle important pour l'évolution de l'écosystème. Cependant, au dessus de deux espèces, ce modèle ne fonctionne plus, puisqu'il se base sur des valeurs parfois décimales, qui n'existent pas dans la nature.
Le développement du code, qui induit une réflexion sur des problématiques concrètes, nous a beaucoup appris sur le langage python et plus généralement sur la programmation, notamment en ce qui concerne :

-La rédaction de fonctions plus complexes qui nécessitent plus d'organisation et d'une réflexion plus poussée avant de se lancer dans son écriture

-L'utilisation de modules tels que matplotlib et pygames principalement, qui nous ont permis d'animer et d'afficher nos programmes

La manipulation de matrices d'une manière non mathématique, en effet celles-ci représentent un terrain, ainsi il n'y a pas besoin de l'inverser ou de la manipuler, il n'y a que les éléments de la matrice qui importent.

Mais également un tel projet apporte beaucoup sur le travail d'équipe, que ce soit dans la répartition des taches ou dans le respect du planning : la coordination était de rigueur car, si un projet sur deux ordinateurs permet d'être plus rapide, il fallait tout de même que les codes soient compatibles une fois assemblés. Au terme de cette étude, on peut donc affirmer que nous avons répondu à toutes les problématiques auxquelles nous nous étions confrontés.

English summary :

To model the evolution of populations of species from an ecosystem, we decided to chose an comparative approach of two models. The first model is based on two differentials equations (established by Lotka and Volterra) describing the evolutions of populations in a simple chain with two species. The stake was to adapt these equations to a discreet model. The second model studies the behavior of each individual according to his state (hunger, age, sex). Here, it was about studying a model running according to "natural laws". We choose to utilize animations and graphics to represent our results. The study of the results will be on two perspectives. The first on will be a study on the importance of parameters within the model. The second one will be a comparison of the two models, to deduce if models correspond and their limits. We also can find population stability in case of an two species ecosystem, but not when there's more than two species.

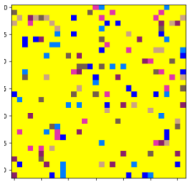
Annexes :

Annexe 1 :

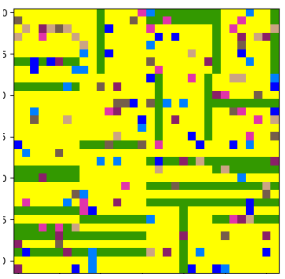
Voici les Terrains initiaux :

Matrice végétale

Matrice animale



La réunion de ces deux matrices avec la fonction Merge_matrix donne ainsi ce résultat :



Les animaux bleus correspondent aux prédateurs « ultimes » (cf la suite), les animaux violets correspondent aux prédateurs intermédiaires et les animaux marrons aux proies.
Chaque couleur a deux nuances, ce qui permet de distinguer les sexes.

Annexe 2 :

Voici le schéma de la fonction cgmt() :
Pour chaque prédateur ultime :

-Si il est trop vieux ou n'a plus d'énergie : Décès
-Si sa jauge d'énergie est trop basse (il a faim) :
-Il cherche la cible la plus proche entre le prédateur intermédiaire et la proie.
-Si cette proie est dans une base adjacente, il la mange
-Sinon, il se rapproche d'une case vers sa cible
-S'il n'a pas faim, qu'il a atteint la maturité sexuelle, et que (si c'est une femelle), elle a fini sa période de gestation, il cherche un animal de son espèce de sexe opposé
-Si cette cible est dans une base adjacente, il se reproduit
-Sinon, il se rapproche d'une case vers sa cible
-S'il n'a rien fait de tout ça, Il se déplace de manière aléatoire
(à chaque tour, son âge augmente, sa jauge d'énergie diminue)

pour chaque prédateur intermédiaire :

-Si il est trop vieux ou n'a plus d'énergie : Décès
-Si sa jauge d'énergie est trop basse (il a faim) :
-Il cherche la proie la plus proche
-Si cette proie est dans une base adjacente, il la mange
-Sinon, il se rapproche d'une case vers sa cible
-S'il n'a pas faim, qu'il a atteint la maturité sexuelle, et que (si c'est une femelle), elle a fini sa période de gestation, il cherche un animal de son espèce de sexe opposé
-Si cette cible est dans une case adjacente, il se reproduit
-Sinon, il se rapproche d'une case vers sa cible
-S'il n'a rien fait, Il se déplace d'une case aléatoirement
(à chaque tour, son âge augmente, sa jauge d'énergie diminue)

pour chaque proie :

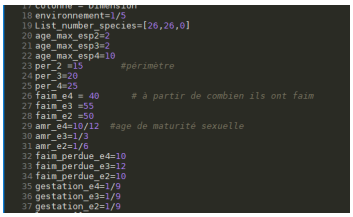
-Si il est trop vieux ou n'a plus d'énergie : Décès
-Si sa jauge d'énergie est trop basse (il a faim) :
-Il cherche la plante la plus proche
-Si cette proie est dans une base adjacente, il la mange
-Sinon, il se rapproche d'une case vers sa cible
S'il n'a pas faim,qu'il a atteint la maturité sexuelle, et que (si c'est une femelle), elle a fini sa période de gestation, il cherche un animal de son espèce de sexe opposé
-Si cette cible est dans une base adjacente, il se reproduit
-Sinon, il se rapproche d'une case vers sa cible
S'il n'a rien fait, Il se déplace d'une case aléatoirement
(à chaque tour, son âge augmente, sa jauge d'énergie diminue)

Enfin, vérification de la jauge d'énergie des plantes pour savoir si elles doivent disparaître ou repousser

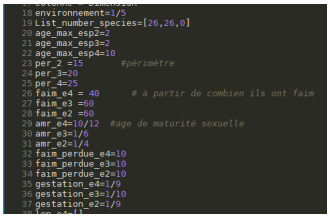
Annexe 3 : Pour chacun des cas, voici les paramètres sélectionnés initialement qui permettent d'observer les résultats analysés (il ne s'agit évidemment pas des seuls paramètres pour tomber sur ce résultat, de plus, il est toujours possible d'avoir un résultat différent en fonction des positions initiales des espèces) :

Pour 2 espèces :

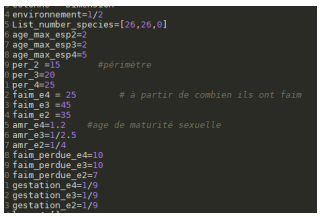
Cas 1 : Victoire des proies
Vidéo



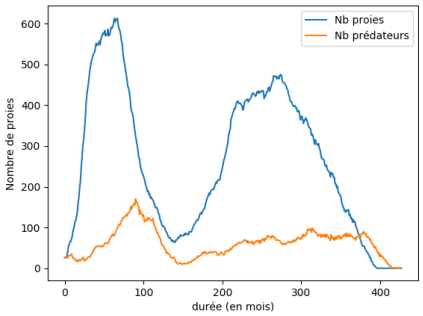
Cas 2 : Victoire des prédateurs
Vidéo



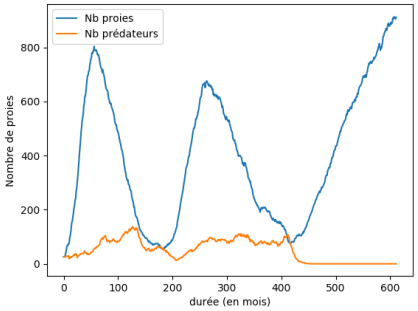
Cas 3 : Victoire dépendante
Vidéos



Pour le Cas 3, voici les courbes de l'évolution de la population :



« Victoire » des prédateurs

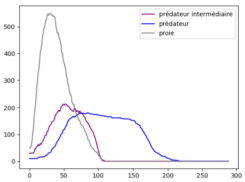


Victoire des proies

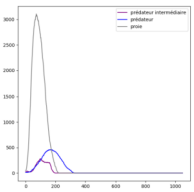
Annexe 4 : Evolution à 3 espèces :
Voici les paramètres sélectionnés :

```
List_number_species=[50,30,10]
age_max_esp2=4
age_max_esp3=8
age_max_esp4=10
per_2 =15       #périmètre
per_3=20
per_4=25
faim_e4 = 40      # à partir de combien ils ont faim
faim_e3 =30
faim_e2 =15
amr_e4=10/12  #age de maturité sexuelle
amr_e3=1/2
amr_e2=1/4
faim_perdue_e4=6
faim_perdue_e3=8
faim_perdue_e2=10
gestation_e4=1/20
gestation_e3=1/20
gestation_e2=1/12
```

Avec ces paramètres, voici les courbes d'évolutions des espèces avec un terrain de taille différente :



Dimension =32



Dimension =64

Annexe 5 : Fonctions utilisées :
Afin de mieux comprendre l'ensemble du programme voici la liste exhaustive des fonctions utilisées :

basea, terrain_base, randomi, terrain_final : Ces fonctions permettent de créer le terrain sans les animaux, l'unique difficulté rencontrée a été de trouver comment ajouter les buissons de végétation, la solution a été de multiplier une matrice avec uniquement des 1 par une matrice diagonale avec comme terme en diagonale soit des 1 soit des 0 en fonction du paramètre environnement. La multiplication de ces deux matrices donne une matrice avec que des 1 sauf certaines bandes, qui sont donc les buissons.

create_empty_land, create_occupied_land, create_occupied_land : Ces fonctions permettent de créer les matrices animales, en fonction du nombre d'animaux voulus par espèce et le nombre d'espèce.

Les fonctions creatdict_ : Permet de créer les dictionnaires des animaux mais aussi des plantes.

Merge_matrix : Réunit les deux matrices végétales et animales en une seule matrice.

Affichage : Transforme chaque case du tableau en une valeur rgb pour l'animer ensuite.

Déplacement : Déplace chaque espèce en utilisant perimetre_vision. Difficulté trouvée : pathfinding, prise en compte des autres animaux pour éviter les collisions. Solution : Utiliser la longueur de chaque case : si la longueur est égale à 1, il n'y a pas d'animal alors que si la longueur est égale à 2, elle est occupée

perimetre_vision : « radar » de chaque animal, qui lui permet de détecter toute cible dans un voisinage de N cases. Difficulté trouvée : Trouver une solution si aucune cible est trouvée. Solution : retourne des coordonnées négatives, ce qui est impossible et facilement contournables en utilisant des conditions

predation : Lorsqu'un prédateur touche une cible et que le prédateur a faim, il mange sa cible et remplit sa jauge de faim

predation_plant : Quand le prédateur se trouve à côté d'une case avec de la nourriture, il mange. La quantité de ressources diminue et la jauge de faim est remplie. Difficulté : Comment réagir si la plante n'a plus d'énergie. Solution : avec la fonction cgmt, la transforme en terre normale pendant un certain nombre de tours avant de repousser

Reproduction : Lorsque deux animaux de même espèce et de sexe différents se touchent, et que la femelle n'est plus en gestation, ils s'accouplent et le nouveau-né apparaît autour de ses parents. S'il n'a pas de place, alors la reproduction a lieu mais il n'y a pas d'enfant.

déplacement_random : déplace l'animal dans une case autour si elle n'est pas occupée.

Proche : A l'aide de la fonction périmetre_vision, renvoie la cible la plus proche entre deux cibles de valeur différentes (est utilisé pour choisir l'espèce la plus proche entre le mâle et la femelle lors d'une prédation)

cgmt : Parcourt tout les dictionnaires et réagit en fonction de chaque animal. Une itération = 1 mois (choix arbitraire).

Update : utilisée pour l'animation.