

Compte rendu du TP4 – Simulation de lapin

Rayan SAKHY
ISIMA – École d'Ingénieurs en Informatique

Janvier 2025

```
Nombre de lapins dans la population au mois 190 : 15547659
Pour 94281840 lapins morts et 109829499 naissances de lapins.
Pour un total de 1986275 lapins matures morts, 4430643 enfants morts et 87864922 lapins juvéniles morts au total.
Pour 7249639 lapins morts et 8444559 naissances de lapins.
Dont 151850 lapins matures, 340811 lapins non matures et 6756978 lapins juvéniles morts ce mois-ci

Nombre de lapins dans la population au mois 191 : 16840615
Pour 102120893 lapins morts et 118961508 naissances de lapins.
Pour un total de 2151210 lapins matures morts, 4799342 enfants morts et 95170341 lapins juvéniles morts au total.
Pour 7839053 lapins morts et 9132009 naissances de lapins.
Dont 164935 lapins matures, 368699 lapins non matures et 7305419 lapins juvéniles morts ce mois-ci

Nombre de lapins dans la population au mois 192 : 18243240
Pour 110623512 lapins morts et 128866752 naissances de lapins.
Pour un total de 2329843 lapins matures morts, 5198915 enfants morts et 103094754 lapins juvéniles morts au total.
Pour 8502619 lapins morts et 9905244 naissances de lapins.
Dont 178633 lapins matures, 399573 lapins non matures et 7924413 lapins juvéniles morts ce mois-ci
```

Table des matières

1	Introduction	5
1.1	Enoncé	5
1.2	Langage choisi	5
2	Choix de modélisation	5
2.1	Intervalle de mise à jour	5
2.2	Taux de mortalité	5
2.3	Probabilité utilisée pour les autres aspects de la simulation	7
2.3.1	Nombre de portées	7
2.3.2	Nombre le lapereaux par portée	7
2.4	Durée de simulation	7
3	Explication du code	7
3.1	La structure du code	7
3.1.1	La classe SimuLapinApplication	8
3.1.2	La classe SimuUtils	8
3.1.3	La classe Lapin	8
3.1.4	La classe LoggerCSV	8
3.2	Certains codes en particulier	8
3.2.1	initHazards()	8
3.2.2	estMortVX(Lapin lapin, double nbAlea)	8
3.2.3	private void miseAJourListe(MersenneTwister rng,LinkedList<Lapin> population)	8
3.2.4	public LinkedList<Lapin> naissance(LinkedList<Lapin> population, MersenneTwister rng)	8
3.2.5	public LoggerCSV(String fileName, int dureeSimu, int numeroSimu)	9
3.2.6	public void logMois(int mois, int vivants, int mortsTot, int mortsBebeTot, int mortsEnfantsTot, int mortsAdulteTot, int naissanceMois, int mortsBebe, int mortsEnfants, int mortsAdulte, int mortsParMois)	9
4	Résultats	9
4.1	Exemple de donnée brute	9
4.2	Données mises sous forme de graphe	10
5	Analyse des résultats	11
5.1	Dynamique globale de la population	11
5.2	Variabilité par tranches (Boxplots)	16
5.3	Moyenne vs Médiane avec bande \pm Écart-type	17
6	Conclusion	18

Table des figures

1	Un exemple des données brutes mises en forme	9
2	Graphe indiquant le nombre de lapins vivants et de son écart type en fonction du temps	10
3	Graphe mettant en avant la relation entre la moyenne et l'écart type du nombre de morts par mois	10
4	Graphe indiquant la proportion de mort juvénile parmi les lapins	11
5	Relation entre le nombre de lapins vivants et le nombre total de morts cumulés .	11
6	Naissances vs Morts par mois et Croissance nette (Δ)	12
7	Cascade annuelle détaillée (Naissances et Morts séparées)	12
8	Cascade annuelle simplifiée (Solde net par année)	13
9	Cascade annuelle sur les 4 premières années (zoom)	13
10	Histogramme empilé des décès par catégorie d'âge (valeurs absolues)	14
11	Histogramme 100% empilé des proportions de décès par catégorie d'âge	14
12	Courbes des proportions de décès par catégorie d'âge	15
13	Variabilité par tranches (boîtes à moustaches) – Vivants.	16
14	Variabilité par tranches – Naissances.	17
15	Moyenne et médiane mensuelles des vivants, avec bande \pm écart-type.	18

Table des Annexes

1	Fonction qui simule des simulations de probabilités en utilisant 2 tableaux	19
2	Fonction qui initialise 2 tableaux contenant les proba de mort pour tous les ages possibles des lapins	19
3	Fonction qui utilise les tableaux initialisés plus tôt	20
4	Fonction qui utilise les fonctions VerifMortVX(...) pour tuer les lapins et les retirer de la liste	21
5	Fonction qui met à jour l'âge et le statut de maturité d'un lapin	21
6	Fonction qui gère les naissances de lapin tous les mois	22
7	Fonction qui crée le fichier et qui ajoute l'entête du fichier dans un StringBuilder	23
8	Fonction qui ajoute les données de chaque mois de simulation dans le StringBuilder	24

1 Introduction

1.1 Enoncé

L'objectif du TP est de concevoir une simulation stochastique à événements discrets pour modéliser la croissance d'une population de lapins de manière réaliste, en tenant compte de la naissance, de la mortalité et du vieillissement. Cette approche est dite individual-based modeling, car chaque lapin est représenté avec ses propres caractéristiques.

Une femelle peut avoir entre 3 et 9 portées par an, avec une probabilité plus élevée pour 5, 6 ou 7. Chaque portée contient 3 à 6 lapereaux, avec une répartition sexuée aléatoire ($\approx 50\%$ mâles/femelles). La maturité sexuelle est atteinte entre 5 et 8 mois.

1.2 Langage choisi

J'ai choisi le langage Java pour pouvoir modéliser des lapins en tant qu'objets. Les avantages du langage à objet pour cette simulation sont la simplicité de modélisation et de modification et sa portabilité grâce à la machine virtuelle java, la JVM.

2 Choix de modélisation

2.1 Intervalle de mise à jour

J'ai choisi de mettre à jour la population de lapins tous les mois pour plus de performance. La mise à jour de la liste de lapin tous les mois permet de limiter le nombre de lapin qu'on ajoute à la liste avant d'appliquer les probabilités de mort, donc de limiter la taille temporaire de la liste.

2.2 Taux de mortalité

Je suis d'abord parti sur les hypothèses de l'énoncé qui concerne les chances de survie et de mort annuelle pour différentes classes d'âges de lapins qui sont :

Age ou stade de vie	Probabilité de vie	Probabilité de mort
Lapereaux	35%	65%
Adultes	60%	40%
10 ans	50%	50%
11 ans	40%	60%
12 ans	30%	70%
13 ans	20%	80%
14 ans	10%	90%
15 ans	0%	100%

Puis j'ai lancé la simulation en faisant un mise à jour tous les mois en utilisant ces probabilités mais mes lapins mourraient trop vite donc j'ai cherché des probabilités de mort plus réalistes.

J'ai trouvé sur ce site internet ([Taux de survie de lapin sauvage](#)) une piste de probabilité par an qui sont :

Age ou stade de vie	Probabilité de vie	Probabilité de mort
Juvéniles (avant 1 mois)	20%	80%
Lapereaux (pas encore mature)	50%	50%
Adultes	80%	20%
8 ans	70%	30%
9 ans	60%	40%
10 ans	50%	50%
11 ans	40%	60%
12 ans	30%	70%
13 ans	20%	80%
14 ans	10%	90%
15 ans	0%	100%

Puis j'ai compris qu'il fallait que je passe les probabilités annuelles en probabilités mensuelles donc les probabilités de morts mensuelles correspondantes au deux choix faits sont :

1. Probabilité dans l'énoncé

- Lapereaux : 6,49%
- Adulte : 4,26%
- 10 ans : 5,61%
- 11 ans : 6,98%
- 12 ans : 8,47%
- 13 ans : 9,97%
- 14 ans : 12,60%
- 15 ans : 100%

2. Probabilité trouvée sur Internet

- Juvénile (avant 1 mois) : 80%
- Lapereaux (pas encore mature) : 5,61%
- Adultes : 1,84%
- 8 ans : 2,96%
- 9 ans : 4,26%
- 10 ans : 5,61%
- 11 ans : 6,98%
- 12 ans : 8,47%
- 13 ans : 9,97%
- 14 ans : 12,60%
- 15 ans : 100%

2.3 Probabilité utilisée pour les autres aspects de la simulation

Pour déterminer le nombre de portées et le nombre de lapereaux par portée j'ai utilisé une méthode pour obtenir des probabilités à partir d'un tableau qui liste les chances de chaque possibilité d'un événement

`tirageProbaSelonPoids(double[] poids,MersenneTwister rng) :`

2.3.1 Nombre de portées

Nombre de portées	Probabilité
3	5%
4	10%
5	20%
6	30%
7	20%
8	10%
9	5%

Cet ensemble de probabilité correspond plus à une gaussienne ce qui ne correspond pas vraiment à l'énoncé mais j'ai trouvé cela plus réaliste.

2.3.2 Nombre le lapereaux par portée

Nombre de portées	Probabilité
3	25%
4	25%
5	25%
6	25%

Pour ce qui est du nombre de lapereaux par portée j'ai mis en place une équiprobabilité.

2.4 Durée de simulation

Pour la durée de simulation j'ai choisi de faire durer la simulation pendant 16 ans, c'est à dire pendant **192 mois** pour 31 itérations de la simulation. Puisque ce sont les limites imposées par mon ordinateur et par mes choix de modélisation.

3 Explication du code

Dans cette partie je vais expliquer la structure et la logique de mon code. En commençant par sa structure.

3.1 La structure du code

Dans mon code il y a 4 classes :

- Une qui exécute le programme
- Une qui stocke un lapin
- Une qui gère un logger pour récupérer les données
- Une qui contient la logique métier de la simulation

3.1.1 La classe SimuLapinApplication

Cette classe initialise Mersenne Twister et utilise la classe **SimuUtils**

3.1.2 La classe SimuUtils

Cette classe permet de gérer les morts et les naissances de ces lapins. Ainsi que l'affichage d'informations dans la console pour le débogage, la simulation d'un mois et la simulation des 16 ans.

3.1.3 La classe Lapin

Cette classe contient toutes les données relatives aux lapins, c'est à dire l'âge, le sexe, l'âge de maturité, si le lapin est mature ou non, le nombre de portées restantes et si la lapine est en période de gestation.

L'information du nombre de portée restante et de la gestation ne concerne que les femelles donc je l'initialise pour les mâles puis je ne change pas sa valeur.

3.1.4 La classe LoggerCSV

Cette classe bufferise des informations sur l'exécution du programme puis les écrit dans un fichier pour réaliser des logs utiles pour l'analyse des données.

3.2 Certains codes en particulier

3.2.1 **initHazards()**

Cette fonction est appelé une seule fois au tout début du programme ce qui permet d'éviter un coût algorithmiques à chaque execution de la vérification de morts des lapins.

3.2.2 **estMortVX(Lapin lapin, double nbAlea)**

Ces 2 fonctions utilise la fonction **initHazards()** à chaque vérification de la mort d'un lapin. Cette méthode permet de n'avoir que des accès en O(1) pour la récupération de la probabilité seuil de mort du lapin en fonction de son age.

3.2.3 **private void miseAJourListe(MersenneTwister rng, LinkedList<Lapin> population)**

Cette fonction met à jour l'age de tous les lapins de listes et et vérifie si ils deviennent mature. Cette fonction reinitialise aussi le nombre de portée restante pour toutes les lapines matures selon leur âges.

3.2.4 **public LinkedList<Lapin> naissance(LinkedList<Lapin> population, MersenneTwister rng)**

Cette fonction vérifie d'abord si il existe un lapin male parmi tous les lapins. Dès que ce lapin est trouvé je crée une nouvelle liste de lapin qui contiendra uniquement les lapins qui sont nés durant le mois.

Pour chaque lapines mature qui peut encore faire des portées je vérifie qu'elle n'ait pas été fécondé le mois dernier. Si elle a été fécondé on ajoute à la liste des nouveaux lapins le nombre de bébé qu'elle met à bas puis je réinitialise le nombre d'enfant en formation.

Si elle n'a pas été fécondé je fixe le nombre d'enfant qu'elle va faire naître le mois prochain et je réduits le nombre de portée quelle peut avoir.

3.2.5 **public LoggerCSV(String fileName, int dureeSimu, int numeroSimu)**

Cette fonction permet de créer un fichier qui a un nom composé du temps de la simulation du time stamp et du numéro de la simulation. Ce nom de fichier permet d'assurer son unicité. De plus j'ajoutes les noms des colonnes dans un StringBuilder/

3.2.6 **public void logMois(int mois, int vivants, int mortsTot, int mortsBebeTot, int mortsEnfantsTot, int mortsAdulteTot, int naissanceMois, int mortsBebe, int mortsEnfants, int mortsAdulte, int mortsParMois)**

Cette fonction est exécuté chaque mois et elle permet d'ajouter au StringBuilder les données de chaque mois.

4 Résultats

4.1 Exemple de donnée brute

Mois	Lapins vivants	Nombre de lapin juvéniles morts au total	Nombre de lapin enfant mort au total	Nombre de lapin adulte mort au total	Total de lapins morts	Nombre de naissance par mois	Nombre de lapin juvéniles morts par mois	Nombre de lapin enfant mort par mois	Nombre de lapin adulte mort par mois	Nombre de lapin mort en 1 mois
1	4	0	0	0	0	0	0	0	0	0
2	5	8	0	0	8	8	0	0	0	8
3	5	8	0	0	8	0	0	0	0	0
4	7	18	0	0	18	12	10	0	0	10
5	7	18	0	0	18	0	0	0	0	0
6	8	27	0	0	27	10	9	0	0	9
7	8	27	0	0	27	0	0	0	0	0
8	11	34	0	0	34	10	7	0	0	7
9	11	34	0	0	34	0	0	0	0	0
10	12	43	0	0	43	10	9	0	0	9
11	13	48	0	0	48	6	5	0	0	5
12	15	52	0	1	53	7	4	0	1	5
13	17	55	0	2	57	6	3	0	1	4
14	18	68	0	2	68	12	11	0	0	11
15	18	72	0	2	74	6	6	0	0	6
16	19	79	0	2	81	8	7	0	0	7
17	21	84	2	2	88	9	5	2	0	7
18	27	95	2	2	99	17	11	0	0	11
19	32	106	3	2	111	17	11	1	0	12
20	33	118	4	2	124	14	12	1	0	13
21	34	125	4	2	131	8	7	0	0	7
22	37	141	6	2	149	21	16	2	0	18
23	37	149	7	2	158	9	8	1	0	9
24	37	168	8	2	178	20	19	1	0	20
25	43	178	9	3	190	18	10	1	1	12

FIGURE 1 – Un exemple des données brutes mises en forme

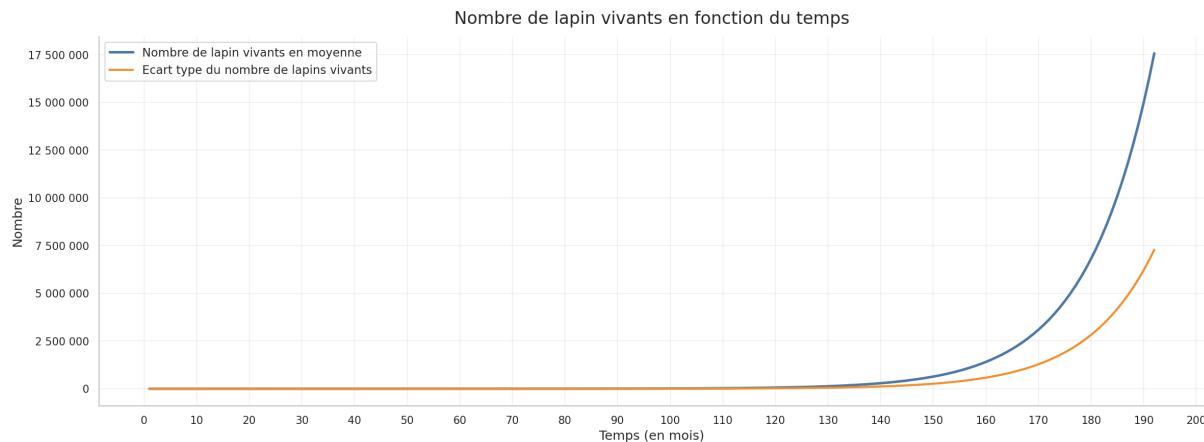


FIGURE 2 – Graphe indiquant le nombre de lapins vivants et de son écart type en fonction du temps

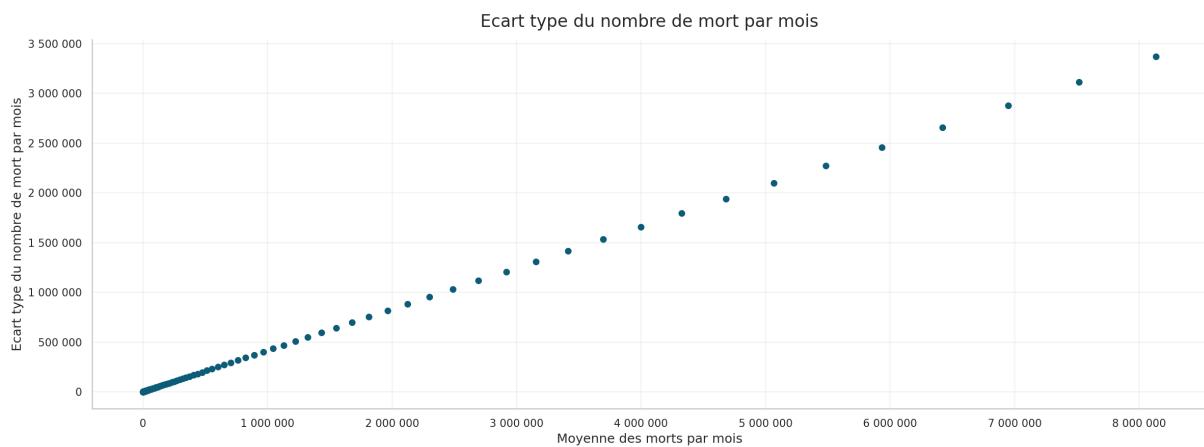


FIGURE 3 – Graphe mettant en avant la relation entre la moyenne et l'écart type du nombre de morts par mois

4.2 Données mises sous forme de graphe

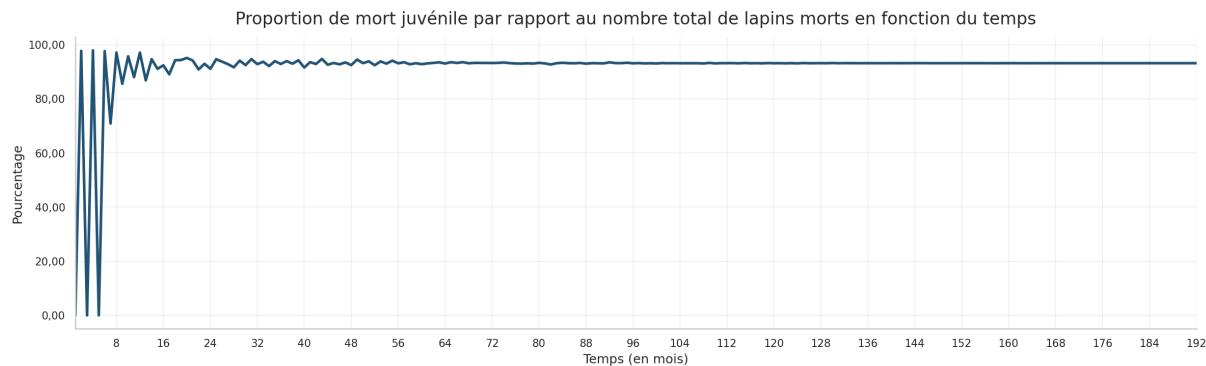


FIGURE 4 – Graphe indiquant la proportion de mort juvénile parmi les lapins

5 Analyse des résultats

5.1 Dynamique globale de la population

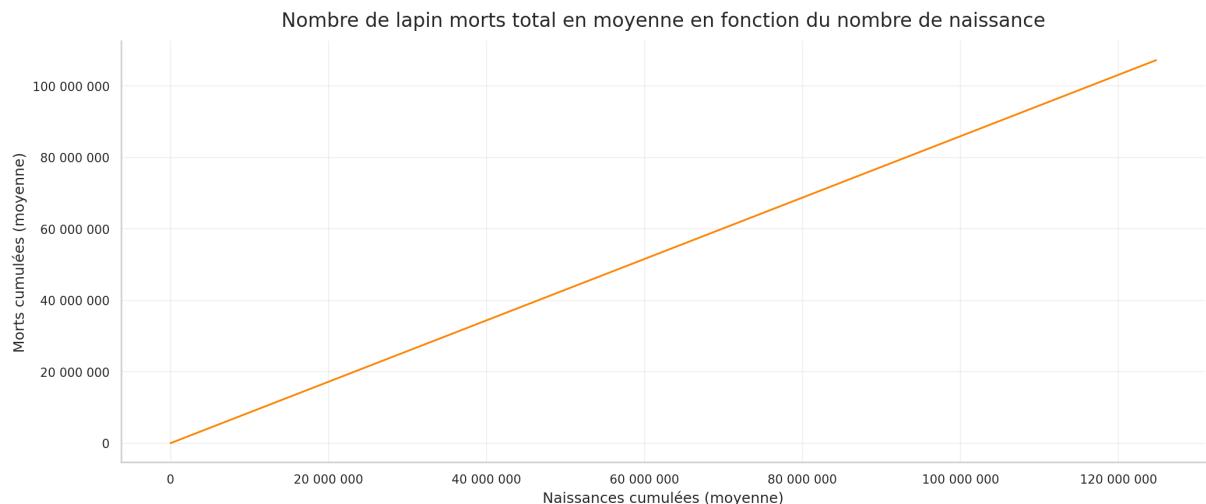
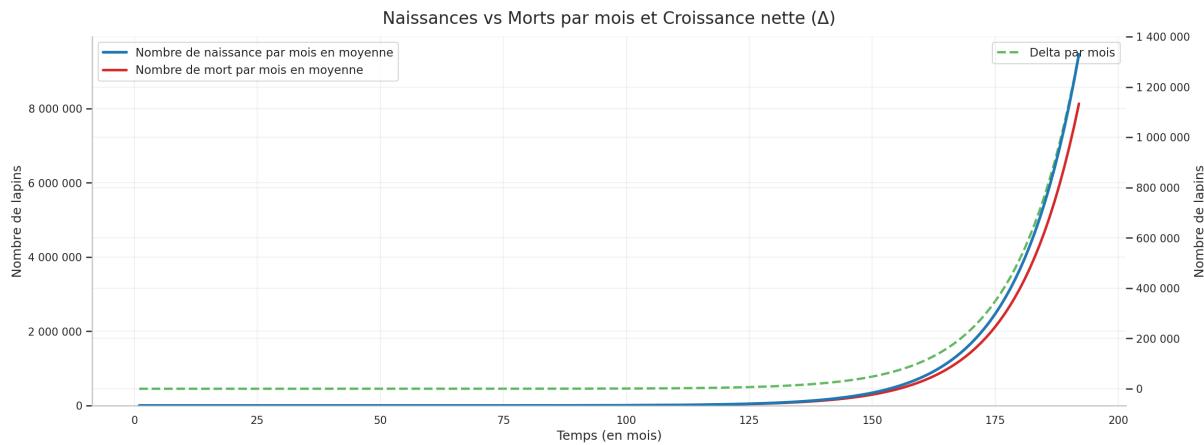
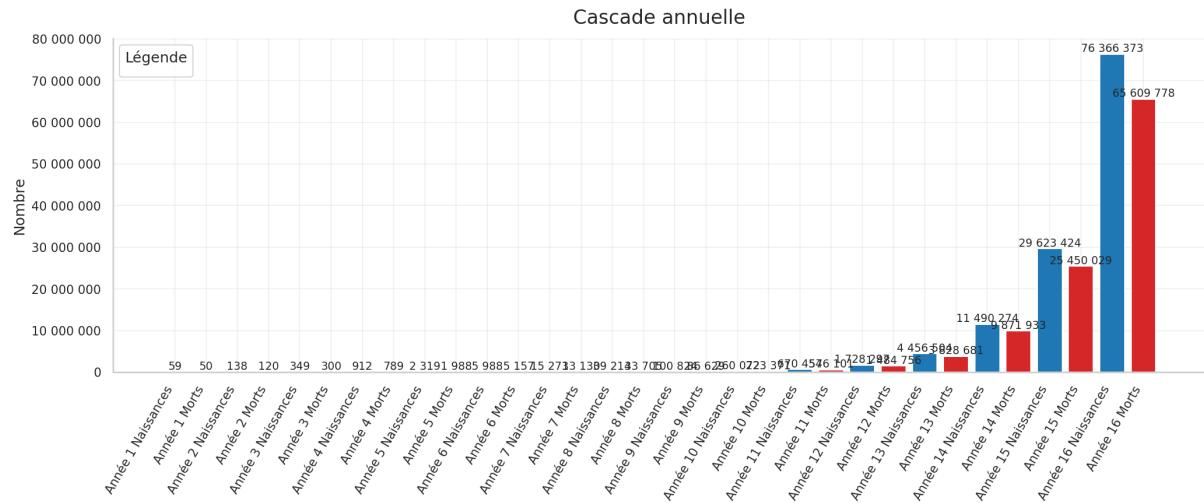


FIGURE 5 – Relation entre le nombre de lapins vivants et le nombre total de morts cumulés

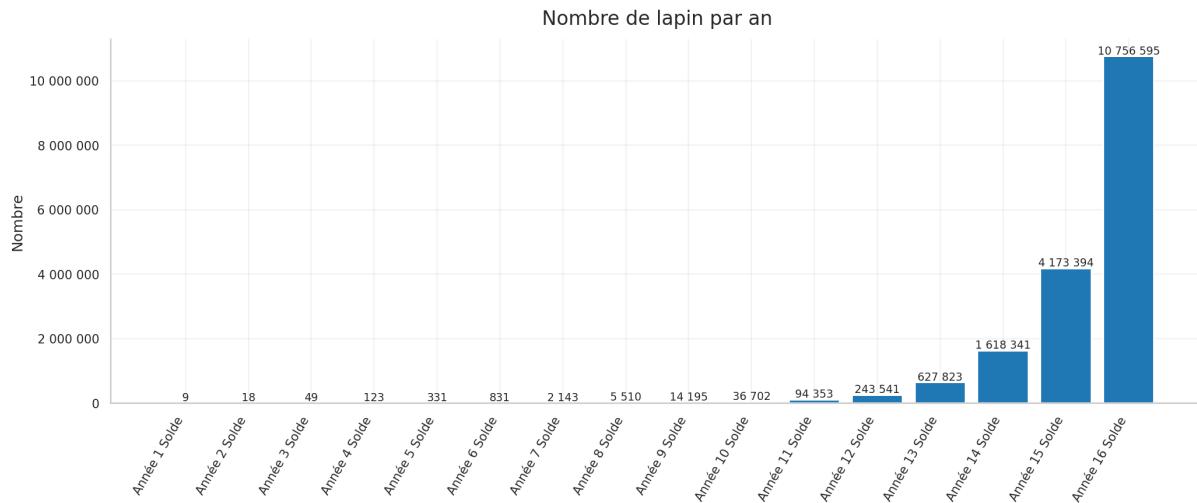
La mortalité cumulée croît de manière exponentielle avec la taille de la population vivante. Cette tendance illustre que plus la population augmente, plus le nombre de décès s'accélère, conséquence directe de la dynamique de reproduction et du vieillissement massif. Les premières phases montrent une croissance lente, mais à partir de plusieurs millions d'individus, la mortalité explose, confirmant l'effet cumulatif des générations.

**FIGURE 6 – Naissances vs Morts par mois et Croissance nette (Δ)**

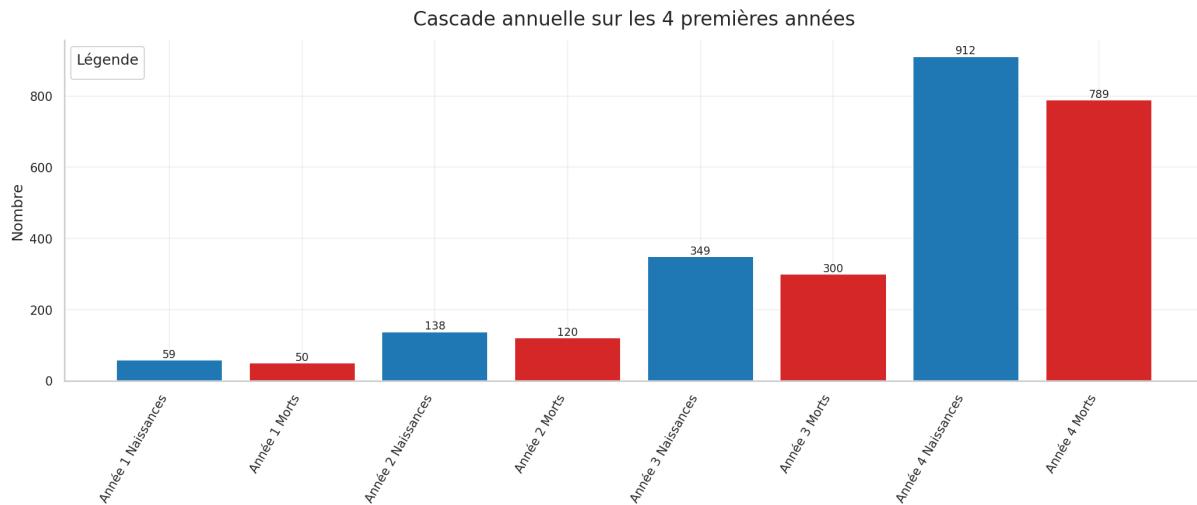
La croissance nette reste positive tout au long de la simulation, mais l'écart entre naissances et morts diminue en fin de période. Cela traduit un ralentissement relatif de la dynamique, probablement lié à la saturation et au vieillissement de la population. Les naissances dominent largement les décès jusqu'à la fin, mais la convergence des courbes suggère une stabilisation à long terme.

**FIGURE 7 – Cascade annuelle détaillée (Naissances et Morts séparées)**

Cette cascade illustre l'accumulation des flux annuels. Les premières années contribuent peu à la croissance, mais à partir de la douzième année, les naissances et morts deviennent massives, dépassant plusieurs dizaines de millions par an. La lisibilité est limitée par l'échelle, mais elle confirme la dynamique exponentielle et l'importance des dernières années dans la population finale.

**FIGURE 8 – Cascade annuelle simplifiée (Solde net par année)**

Le solde annuel reste positif chaque année, confirmant une croissance continue. Après l'année 12, la dynamique s'emballe, avec un gain net supérieur à 15 millions en année 16. Ce graphique est plus lisible que la version détaillée et met en évidence la tendance globale.

**FIGURE 9 – Cascade annuelle sur les 4 premières années (zoom)**

Le zoom sur les premières années montre une croissance lente et équilibrée. Les naissances dépassent légèrement les morts, mais la dynamique reste modérée avant la phase exponentielle. Cette vue est utile pour analyser la phase initiale sans être écrasée par les valeurs extrêmes des dernières années.

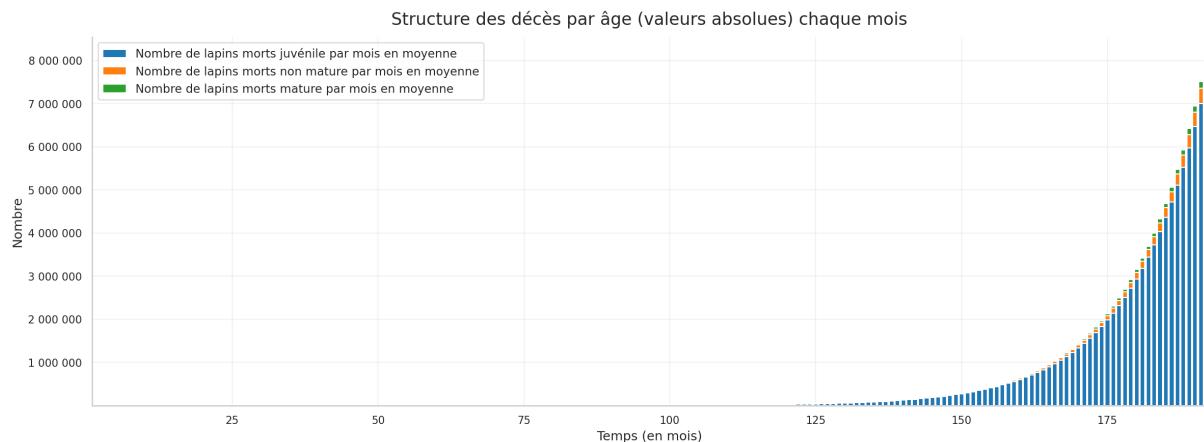


FIGURE 10 – Histogramme empilé des décès par catégorie d'âge (valeurs absolues)

Cet histogramme empilé illustre la répartition des décès par catégorie d'âge (juvéniles, non matures, matures) au fil des mois.

Observation : La mortalité juvénile domine largement (>90%) tout au long de la simulation. Les décès non matures apparaissent après environ 5 mois, mais restent faibles. Les décès matures sont quasi inexistant avant 8 ans.

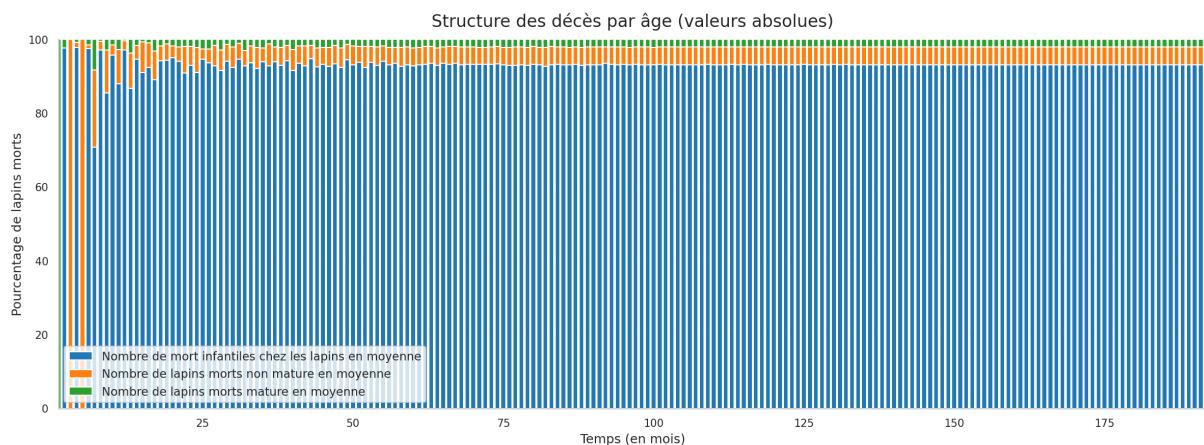


FIGURE 11 – Histogramme 100% empilé des proportions de décès par catégorie d'âge

Ce graphique montre la part relative de chaque catégorie dans les décès totaux.

Observation : La proportion de décès juvéniles se stabilise entre 92% et 95% après environ 40 mois. Les non matures représentent environ 4 à 5%, tandis que les matures restent inférieurs à 1%.

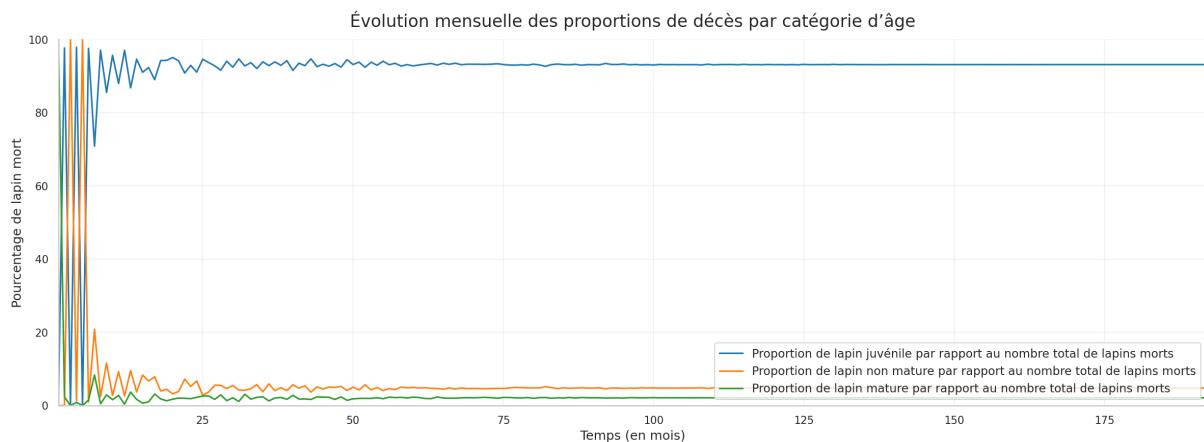


FIGURE 12 – Courbes des proportions de décès par catégorie d’âge

Ce graphique en courbes rend visible les petites proportions.

Observation : Les proportions se stabilisent rapidement :

- juvéniles autour de 93%
- non matures environ 5%
- matures inférieurs à 1%.

Cette représentation est utile pour analyser les tendances fines qui sont peu visibles dans les histogrammes empilés.

5.2 Variabilité par tranches (Boxplots)

Cette sous-section analyse la dispersion des indicateurs clés (vivants, naissances) et la stabilité des tendances au cours des 16 années simulées. L'objectif est d'évaluer la robustesse du modèle et la variabilité inter-mois et inter-simulations. Pour éviter la surcharge visuelle (192 mois), les données ont été regroupées en quatre tranches annuelles : Années 1–4, 5–8, 9–12, 13–16.

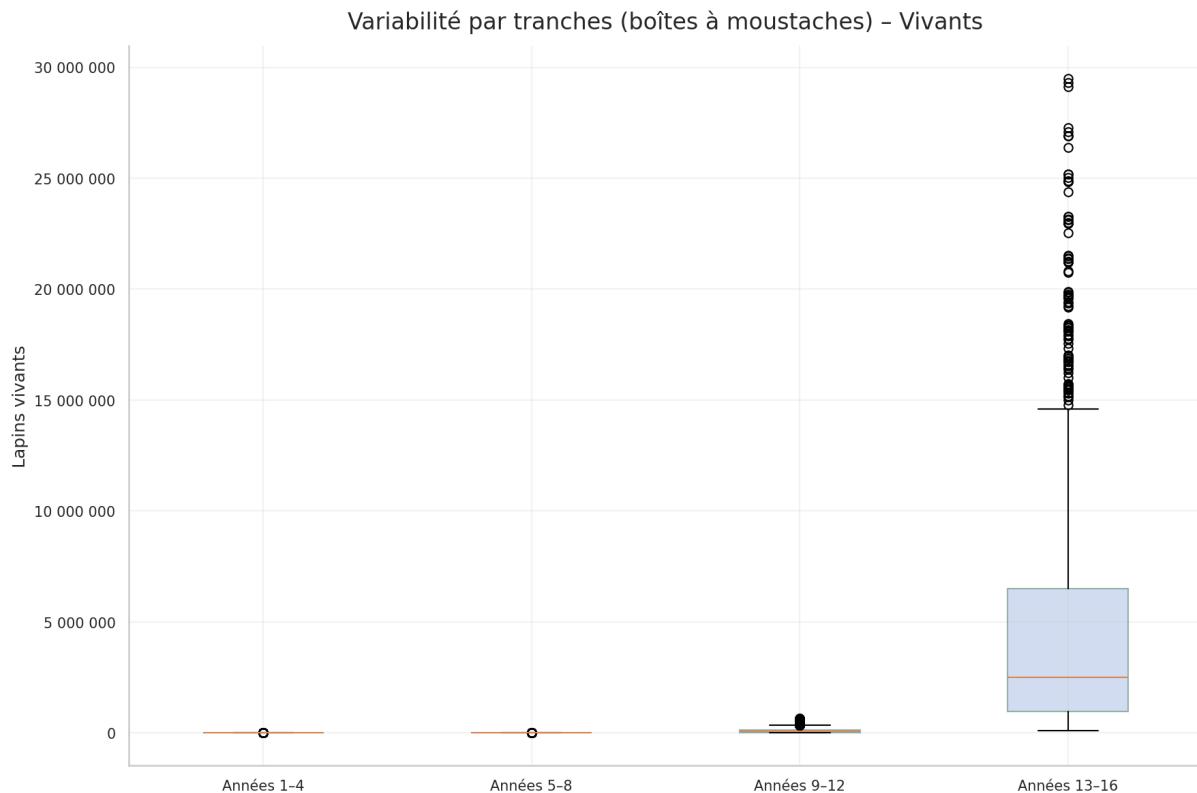


FIGURE 13 – Variabilité par tranches (boîtes à moustaches) – Vivants.

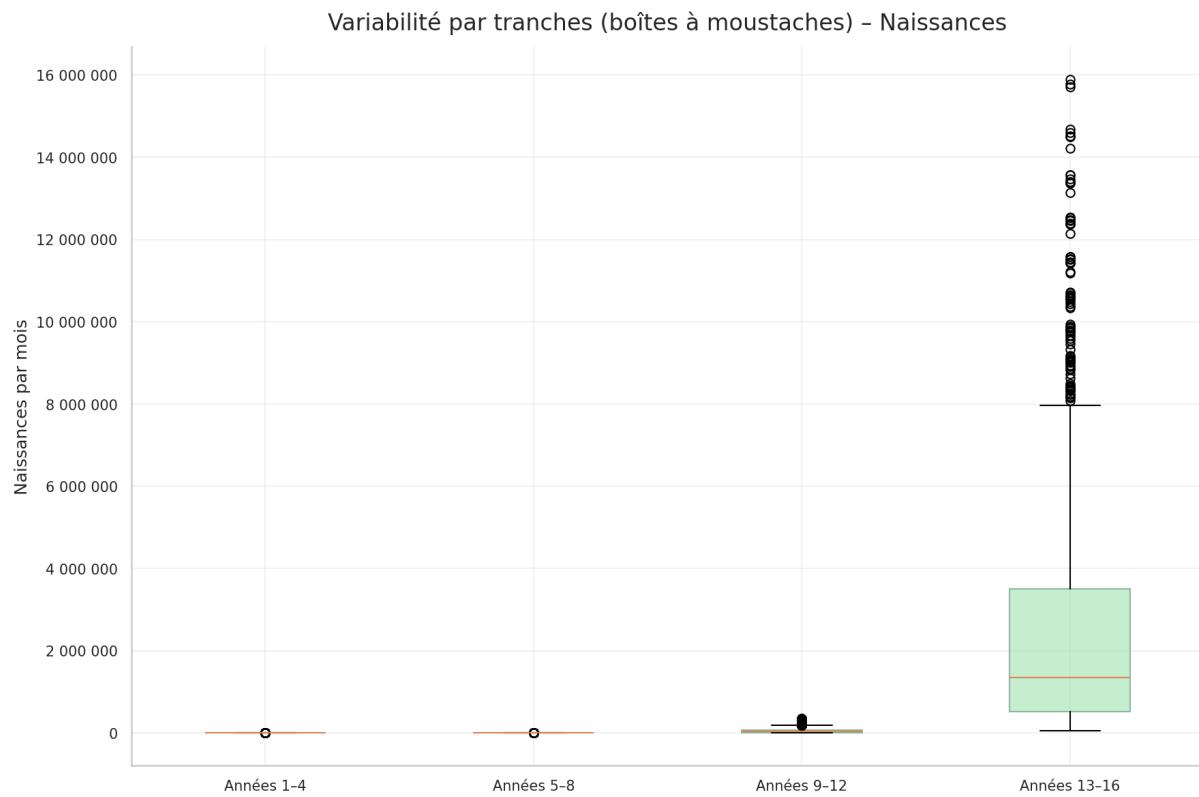


FIGURE 14 – Variabilité par tranches – Naissances.

5.3 Moyenne vs Médiane avec bande \pm Écart-type

La figure illustre l'évolution mensuelle du nombre de lapins vivants en comparant la moyenne et la médiane, tout en intégrant une bande correspondant à l'écart-type. On observe que la moyenne et la médiane restent proches, ce qui indique une distribution relativement symétrique des valeurs au fil du temps. La bande \pm écart-type met en évidence la variabilité : elle s'élargit progressivement, traduisant une dispersion croissante liée à l'augmentation de la population et aux effets stochastiques (naissances aléatoires). Cette représentation permet de juger à la fois de la tendance centrale et de la robustesse du modèle face aux fluctuations.

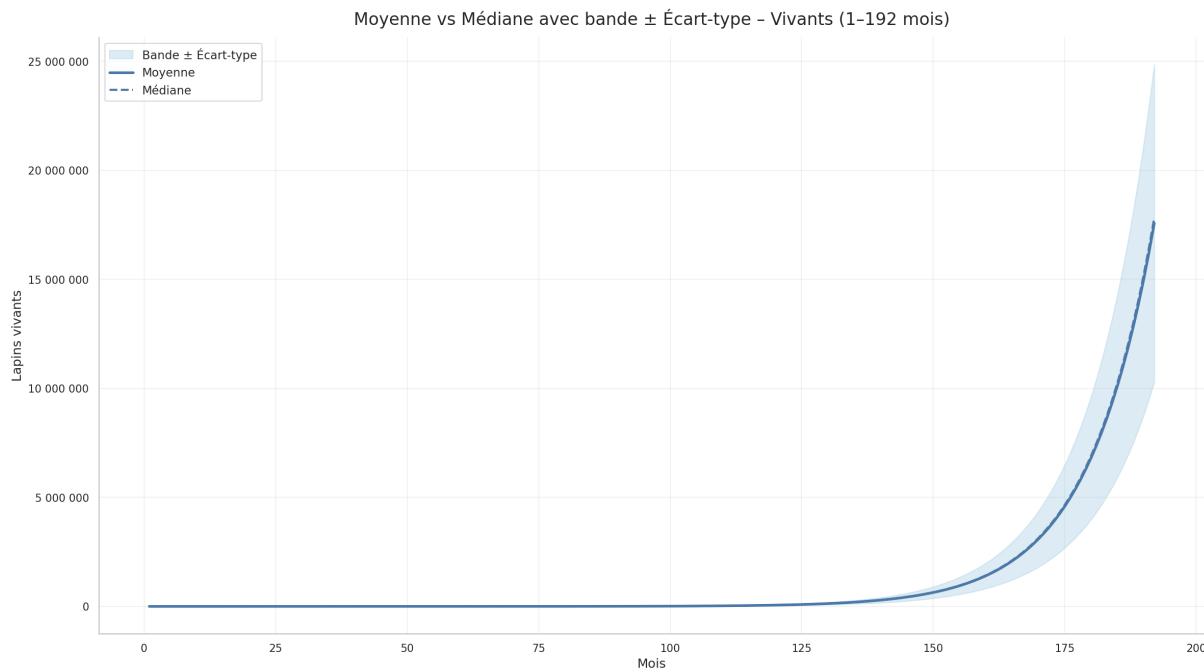


FIGURE 15 – Moyenne et médiane mensuelles des vivants, avec bande \pm écart-type.

6 Conclusion

Cette étude a permis de modéliser et d’analyser la dynamique d’une population de lapins sur une période de 16 ans (192 mois) à l’aide d’une simulation stochastique. Le modèle intègre des paramètres biologiques réalistes tels que la fécondité, la maturité sexuelle, et des probabilités de survie différencierées par âge, ce qui offre une représentation crédible des processus démographiques.

Les résultats montrent une croissance rapide et soutenue de la population, avec un solde net positif tout au long de la période, bien que l’écart entre naissances et décès tende à se réduire en fin de simulation en raison du vieillissement et de la saturation. L’analyse de la structure des décès révèle une mortalité juvénile dominante ($>90\%$), confirmant l’impact critique des premières phases de vie sur la dynamique globale.

L’étude de la variabilité met en évidence une dispersion croissante des effectifs et des flux au fil du temps, mais sans rupture majeure : la proximité entre moyenne et médiane et la stabilité des tendances centrales attestent de la robustesse du modèle face aux aléas stochastiques.

Ces observations soulignent que, malgré une forte sensibilité aux paramètres biologiques, la dynamique globale reste prévisible et cohérente. Ce travail constitue une base solide pour des analyses complémentaires, telles que des études de sensibilité ou des scénarios à long terme, afin d’évaluer l’impact de variations paramétriques sur la croissance et la stabilité de la population.

Annexe

```

1 private static int tirageProbaSelonPoids(double[] poids, MersenneTwister rng) {
2     double total = 0;
3     double nbAlea;
4     double cumul = 0;
5     int i;
6
7     for (double p : poids) {
8         total += p;
9     }
10
11    nbAlea = rng.nextDouble() * total;
12
13    for (i = 0; i < poids.length; i++) {
14        cumul += poids[i];
15        if (nbAlea < cumul) {
16            return i;
17        }
18    }
19    return poids.length - 1;
20 }
```

Code 1 – Fonction qui simule des simulations de probabilités en utilisant 2 tableaux

```

1 private static void initHazards() {
2     for (int age = 0; age <= MAX_AGE; age++) {
3         hazardMatureV1[age] = (age == 180) ? 1.0
4             : (age >= 168) ? 0.126
5             : (age >= 156) ? 0.0997
6             : (age >= 144) ? 0.0847
7             : (age >= 132) ? 0.0698
8             : (age >= 120) ? 0.0561
9             : 0.0426;
10
11        hazardNonMatureV1[age] = 0.0649;
12        hazardMatureV2[age] = (age == 180) ? 1.0
13            : (age >= 168) ? 0.126
14            : (age >= 156) ? 0.0997
15            : (age >= 144) ? 0.0847
16            : (age >= 132) ? 0.0698
17            : (age >= 120) ? 0.0561
18            : (age >= 108) ? 0.0426
19            : (age >= 96) ? 0.0296
20            : 0.0184;
21
22    hazardNonMatureV2[age] = (age < 1) ? 0.8 : 0.0561;
23 }
```

Code 2 – Fonction qui initialise 2 tableaux contenant les proba de mort pour tous les ages possibles des lapins

```

1 private boolean estMortV1(Lapin lapin, double nbAlea) {
2     int a = Math.min(lapin.age, MAX_AGE);
3
4     final double h = lapin.estMature ? hazardMatureV1[a] : hazardNonMatureV1[a];
5
6     boolean mort = (nbAlea < h);
7
8     if (!mort && lapin.estMature && lapin.age >= MAX_AGE) {
9         mort = true;
10    }
11
12    if (mort) {
13        if (lapin.estMature) {
14            nbAdulteMort++;
15            nbAdulteMortTot++;
16        } else {
17            nbEnfantMort++;
18            nbEnfantMortTot++;
19        }
20    }
21
22    return mort;
23 }
24
25 private boolean estMortV2(Lapin lapin, double nbAlea) {
26     int a = Math.min(lapin.age, MAX_AGE);
27     final double h = lapin.estMature ? hazardMatureV2[a] : hazardNonMatureV2[a];
28     boolean mort = (nbAlea < h);
29     if (!mort && lapin.estMature && lapin.age >= MAX_AGE) {
30         mort = true;
31     }
32     if (mort) {
33         if (lapin.estMature) {
34             nbAdulteMort++;
35             nbAdulteMortTot++;
36         } else if (lapin.age < 1) {
37             nbBebeMort++;
38             nbBebeMortTot++;
39         } else {
40             nbEnfantMort++;
41             nbEnfantMortTot++;
42         }
43     }
44     return mort;
45 }
```

Code 3 – Fonction qui utilise les tableaux initialisés plus tôt

```

1 public LinkedList<Lapin> verifMort(LinkedList<Lapin> population,
2     MersenneTwister rng) {
3     // System.out.println("test entre mort : " +l.get(3).age);
4     LinkedList<Lapin> survivants = new LinkedList<>();
5     for (Lapin lapin : population) {
6         if (!estMortV2(lapin, rng.nextDouble())) {
7             survivants.add(lapin);
8         } else {
9             nbLapinMort++;
10            nbLapinMortTot++;
11        }
12    }
13    // System.out.println("taille liste l : "+l.size());
14    return survivants;
15 }
```

Code 4 – Fonction qui utilise les fonctions VerifMortVX(...) pour tuer les lapins et les retirer de la liste

```

1 private void miseAJourListe(MersenneTwister rng, LinkedList<Lapin> population)
2 {
3     for (Lapin lapin : population) {
4         if (lapin.sex == 1 && lapin.age % 12 == 0) {
5             lapin.nbPorteeRestante = nombrePortee(rng);
6         }
7
8         lapin.age += 1;
9
10        if (!lapin.estMature) {
11            if (lapin.age >= 8) {
12                lapin.estMature = true;
13            } else if (lapin.age >= lapin.ageMaturite) {
14                lapin.estMature = true;
15            }
16        }
17    }
18 }
```

Code 5 – Fonction qui met à jour l'âge et le statut de maturité d'un lapin

```

1 public LinkedList<Lapin> naissance(LinkedList<Lapin> population,
2     MersenneTwister rng) {
3     boolean malePresent = false;
4     Lapin currentLapin;
5
6     Iterator<Lapin> it = population.iterator();
7     while (it.hasNext() && !malePresent) {
8         currentLapin = it.next();
9         if (currentLapin.sex == 0 && currentLapin.estMature) {
10             malePresent = true;
11         }
12     }
13     miseAJourListe(rng, population);
14
15     if (!malePresent) {
16         return population;
17     }
18
19     double[] poids = {0.25, 0.30, 0.45};
20     int[] possibilite = {5, 6, 7};
21
22     LinkedList<Lapin> nouveaux = new LinkedList<>();
23     for (Lapin lapin : population) {
24
25         if (lapin.sex == 1 && lapin.estMature && lapin.nbPorteeRestante > 0) {
26             if (lapin.nbEnfantEnGestation == 0) {
27                 lapin.nbPorteeRestante -= 1;
28                 lapin.nbEnfantEnGestation = nombreEnfants(rng);
29             } else {
30                 for (int i = 0; i < lapin.nbEnfantEnGestation; i++) {
31                     // sexe aléatoire 0/1, ageMaturite tiré une fois à la naissance
32                     int sexe = rng.nextInt(2);
33                     int ageMaturite = possibilite[tirageProbaSelonPoids(poids, rng)];
34                     Lapin bebe = new Lapin(0, sexe, ageMaturite);
35                     nouveaux.add(bebe);
36                     nbLapinTot++;
37                     nbLapin++;
38                 }
39                 lapin.nbEnfantEnGestation = 0;
40             }
41         }
42     }
43     nouveaux.addAll(population);
44     return nouveaux;
45 }
```

Code 6 – Fonction qui gère les naissances de lapin tous les mois

```
1 public LoggerCsv(String fileName, int dureeSimu, int numeroSimu) {
2     sb = new StringBuilder();
3
4     try {
5         String timestamp = LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd/MM HH:mm"));
6         fileName += "_" + dureeSimu + "_mois_" + timestamp + "_" + numeroSimu + ".csv";
7         writer = new FileWriter(fileName);
8
9         // writer.write("Timestamp : " + timestamp + "\n");
10        sb.append("Timestamp : ");
11        sb.append(timestamp);
12        sb.append("\n");
13
14        // writer.write("Mois,Lapins vivants,Nombre de lapin juvéniles morts au total,Nombre de lapin enfant mort au total,Nombre de lapin adulte mort au total,Total de lapins morts,Nombre de naissance par mois,Nombre de lapin juvéniles morts par mois,Nombre de lapin enfant mort par mois,Nombre de lapin adulte mort par mois,Nombre de lapin mort en 1 mois\n");
15        sb.append("Mois,Lapins vivants,Nombre de lapin juvéniles morts au total,Nombre de lapin enfant mort au total,Nombre de lapin adulte mort au total,Total de lapins morts,Nombre de naissance par mois,Nombre de lapin juvéniles morts par mois,Nombre de lapin enfant mort par mois,Nombre de lapin adulte mort par mois,Nombre de lapin mort en 1 mois\n");
16    } catch (IOException e) {
17        e.printStackTrace();
18    }
19 }
```

Code 7 – Fonction qui crée le fichier et qui ajoute l'entête du fichier dans un StringBuilder

```
1 public void logMois(int mois, int vivants, int mortsTot, int mortsBebeTot, int
2     mortsEnfantsTot, int mortsAdulteTot, int naissanceMois, int mortsBebe, int
3     mortsEnfants, int mortsAdulte, int mortsParMois) {
4     // try {
5     // StringBuilder sb = new StringBuilder();
6
7     sb.append(mois);
8     sb.append(",");
9     sb.append(vivants);
10    sb.append(",");
11    sb.append(mortsBebeTot);
12    sb.append(",");
13    sb.append(mortsEnfantsTot);
14    sb.append(",");
15    sb.append(mortsAdulteTot);
16    sb.append(",");
17    sb.append(mortsTot);
18    sb.append(",");
19    sb.append(naissanceMois);
20    sb.append(",");
21    sb.append(mortsBebe);
22    sb.append(",");
23    sb.append(mortsEnfants);
24    sb.append(",");
25    sb.append(mortsAdulte);
26    sb.append(",");
27    sb.append(mortsParMois);
28    sb.append("\n");
29
30    // writer.write(mois + "," + vivants + "," + mortsBebeTot + "," +
31    //   mortsEnfantsTot + "," + mortsAdulteTot + "," + mortsTot + "," +
32    //   naissanceMois + "," + mortsBebe + "," + mortsEnfants + "," + mortsAdulte +
33    //   "," + mortsParMois + "\n");
34 } catch (IOException e) {
35     e.printStackTrace();
36 }
```

Code 8 – Fonction qui ajoute les données de chaque mois de simulation dans le StringBuilder