Simulation TP4

Étude de l'évolution de la population chez les lapins

Auteur : Thibault VERMEULEN

Encadrant : David HILL

Année universitaire : 2025–2026

ISIMA — INP

Table des matières

Table des figures

1 Introduction

L'objectif de ce TP est de manipuler et comparer différents générateurs de nombres aléatoires pour simuler la croissance d'une population de lapins. Par le biais d'expériences numériques, nous chercherons à produire des trajectoires de population « réalistes » et à identifier les modèles stochastiques et les réglages qui influencent le comportement simulé.

Ce travail permet d'apprendre deux choses : d'une part, acquérir une maîtrise pratique des générateurs aléatoires et de leur implémentation ; d'autre part, comprendre l'impact du choix du générateur (ainsi que de ses paramètres) sur la fidélité des simulations.

2 Exercice préliminaire

Avant de chercher à modéliser la croissance d'une population de lapins à l'aide de modèles stochastiques plus complexes, il est pertinent de débuter par une approche simple et déterministe. Nous allons donc nous appuyer sur le modèle historique proposé par Leonardo Fibonacci, qui illustre de manière élémentaire la dynamique d'une population de lapins en fonction du temps.

Ce modèle repose sur une hypothèse idéalisée : un couple de lapins atteint la maturité reproductive au bout d'un mois et engendre un nouveau couple chaque mois à partir de ce moment, sans mortalité ni limitation de ressources. Cette formulation conduit naturellement à la célèbre suite de Fibonacci, où chaque terme correspond au nombre total de couples à un instant donné.

2.0.1 Graphe en échelle normale

Le graphique ci-dessous illustre l'évolution de la population calculée à l'aide de ce modèle sur une période de huit années, avec un pas de temps mensuel. Il met en évidence la croissance exponentielle caractéristique du modèle, qui bien qu'irréaliste dans un cadre biologique, constitue une base conceptuelle utile pour introduire les modèles aléatoires étudiés par la suite.

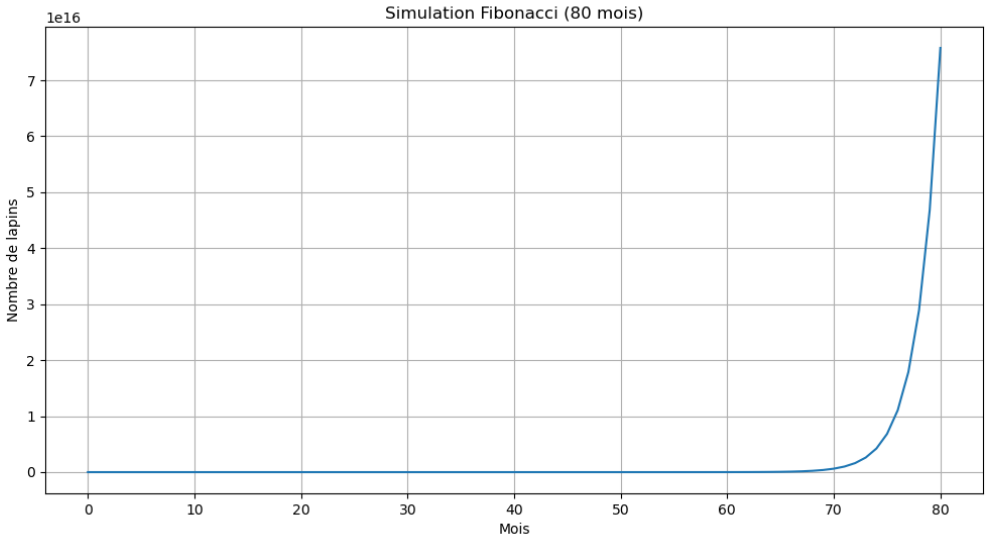

FIGURE 1 – Population selon le modèle de Fibonacci

On peut clairement observer que la croissance de la population suit une tendance exponentielle. Pour mieux analyser cette évolution, il est intéressant de représenter l'axe des ordonnées en échelle logarithmique.

2.0.2 Graphe en échelle logarithmique

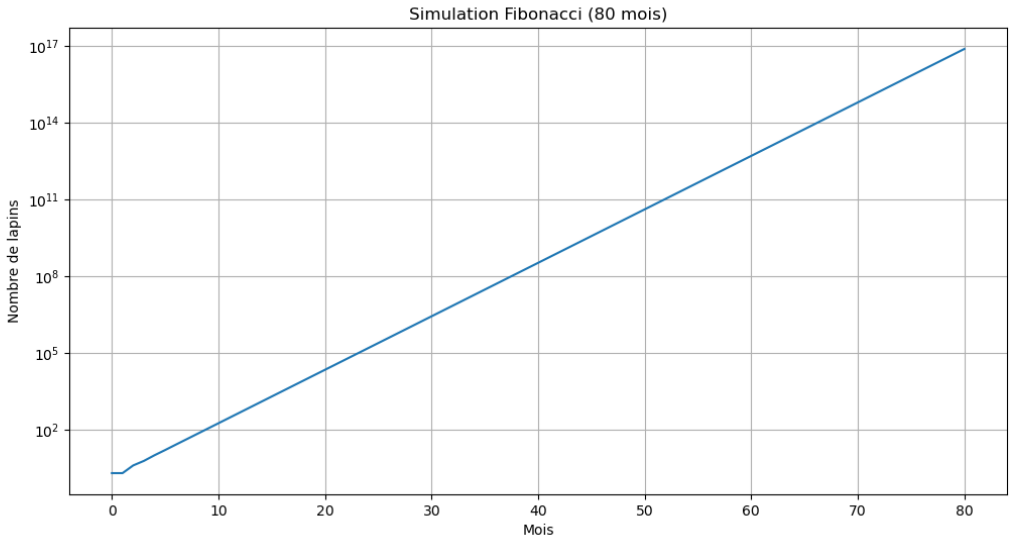

FIGURE 2 – Population selon le modèle de Fibonacci avec log

En représentant l'axe des ordonnées en échelle logarithmique, on constate que la croissance de la population apparaît désormais quasi linéaire, ce qui met en évidence la nature exponentielle du phénomène.

3 Le projet

Après avoir réalisé l'exercice préliminaire, on constate que l'utilisation du modèle basé sur la suite de Fibonacci conduit à des résultats peu réalistes. C'est pourquoi nous proposons maintenant de développer un modèle permettant de représenter l'évolution d'une population de lapins de manière plus fiable.

3.1 Mise en contexte du modèle principal

Contrairement au modèle déterministe précédent, ce modèle est *stochastique* et *individuel* : chaque lapin est considéré individuellement et possède ses propres paramètres qui influencent sa survie et sa reproduction.

Le modèle prendra en compte plusieurs aspects importants :

- la mortalité, avec des probabilités de survie différentes selon l'âge des lapins ;
- le vieillissement, afin de suivre les naissances et les décès au fil du temps ;
- la reproduction, avec un nombre de portées annuel variable et un nombre de petits par portée aléatoire ;
- le sexe des nouveaux individus, déterminé aléatoirement avec une probabilité proche de 50 %.

L'objectif est de simuler l'évolution de la population de manière plus fidèle aux dynamiques biologiques naturelles, tout en conservant une approche simple au départ. Les paramètres et probabilités utilisés seront définis de manière réaliste, tout en restant accessibles pour l'implémentation.

3.2 Méthodologie

Pour commencer, on répartit les lapins dans quatre tableaux distincts : jeunes mâles, jeunes femelles, mâles adultes et femelles adultes. Cette organisation permet d'optimiser le parcours des lapins, en ne traitant à chaque étape que les individus concernés par les opérations de vieillissement, de mortalité ou de reproduction.

La méthodologie suivie peut être résumée en plusieurs étapes principales :

1. Initialisation de la population

La simulation commence par la création des lapins de départ, séparés en jeunes mâles et jeunes femelles. Chaque lapin voit son âge (en mois) initialisé à 0, et son sexe est déterminé aléatoirement selon une probabilité donnée. La maturité sexuelle des lapins est à ce moment là, tirée de manière aléatoire selon une distribution gaussienne, pour représenter au mieux les différences qu'on peut avoir entre chaque individu.

Pour les femelles, un calendrier annuel de portées est défini pour répartir leurs naissances sur l'année.

2. Mise à jour mensuelle de l'état des lapins

Pour chaque mois de simulation, chaque lapin subit un vieillissement et une vérification de survie selon sa catégorie d'âge :

- Les jeunes lapins (pré-adultes) ont une probabilité de mortalité plus élevée que les adultes.
- Les adultes peuvent mourir selon un taux qui augmente avec l'âge.
- Les lapins atteignant leur maturité sont transférés des tableaux de jeunes vers ceux des adultes.

3. Reproduction

Chaque femelle adulte peut avoir des portées selon le calendrier fixé tous les 12 mois (et chaque femelle a son propre calendrier). La reproduction ne se produit que s'il existe au moins un mâle adulte dans la population. Le nombre de petits par portée est tiré aléatoirement, et leur sexe est déterminé par tirage aléatoire avec une probabilité de 50 %. Les nouveaux lapins sont ajoutés aux tableaux correspondants de jeunes mâles et jeunes femelles.

4. Suivi de la population

Après la mise à jour des états et des naissances, le nombre total de lapins vivants est enregistré pour le mois courant, permettant de visualiser l'évolution de la population au fil du temps.

5. Fin de la simulation et libération de la mémoire

Après avoir parcouru tous les mois de simulation, la mémoire allouée pour chaque lapin et pour les tableaux de population est libérée afin d'éviter les fuites mémoire.

Cette méthodologie permet de suivre de manière détaillée l'évolution d'une population de lapins en intégrant le vieillissement, la mortalité, la reproduction et la dynamique individuelle de chaque lapin, tout en conservant une approche stochastique réaliste.

3.3 Générateur aléatoire utilisé

La simulation repose sur le MersenneTwister, qui est utilisé à plusieurs niveaux :

- Détermination du sexe des lapins à la naissance (probabilité d'être femelle ou mâle) ;
- Détermination de l'âge de maturité sexuelle selon une distribution gaussienne (5 à 8 mois) ;
- Nombre de portées annuelles d'une femelle (distribution gaussienne 3 à 9 portées) ;
- Nombre de petits par portée (tirage uniforme entier entre 3 et 6) ;
- Évaluation de la mortalité mensuelle selon l'âge et les probabilités associées.
- Mélange des mois de naissance des portées d'une femelle (algorithme de Fisher-Yates partiel).

Utiliser ce générateur aléatoire permet d'avoir des tirages de haute qualité statistique pour les événements biologiquement sensibles.

3.4 Implémentation

Pour mettre en œuvre le MersenneTwister, plusieurs fonctions ont été codées. Les deux principales sont les suivantes :

- **Générateur uniforme entier :**

```c
int randomUniformeEntreAEtBEntier(int a, int b) {
    return a + (int)((b - a + 1) * genrand_real1());
}
```

  Cette fonction produit un entier aléatoire dans un intervalle donné. Elle sert par exemple à déterminer le nombre de petits par portée.

- **Générateur gaussien tronqué :**

```c
int gaussianEntreAetB(double a, double b) {
    double u1, u2, z;
    u1 = genrand_real1();
    u2 = genrand_real1();
    z = sqrt(-2.0 * log(u1)) * cos(2.0 * M_PI * u2);
    double value = (a+b)/2 + z;
    if (value < a) value = a;
    if (value > b) value = b;
    return (int) value;
}
```

Cette fonction applique la transformation de Box-Muller pour générer des valeurs suivant une loi normale, tronquées dans l'intervalle $[a, b]$. Elle est utilisée pour déterminer l'âge de maturité sexuelle et le nombre de portées annuelles.

— **Répartition des portées avec Fisher-Yates :**

```
for (int i = 0; i < nbPortee; i++) {
    int j = (int)(i + tabMersenneTwister[*index] % (12 - i));
    *index = (*index + 1) % tailleTabMersenneTwister;

    int tmp = repartitionPortee[i];
    repartitionPortee[i] = repartitionPortee[j];
    repartitionPortee[j] = tmp;
}
```

Cette étape utilise des nombres aléatoires stockés dans **tabMersenneTwister** initialisé au début de l'expérience, pour répartir aléatoirement les portées d'une femelle sur les 12 mois, évitant ainsi des schémas reproductifs réguliers.

Ces extraits montrent comment le code traduit les concepts stochastiques en opérations concrètes.

Le reste du code source sera disponible dans les annexes.

3.5 Optimisation

Au départ, l'exécution de mon programme était relativement lente, avec un temps d'environ 30 secondes pour le traitement de 96 mois. Après analyse des performances, j'ai constaté que la principale source de ralentissement provenait de l'algorithme de mélange Fisher-Yates. En effet, la fonction **nbEnfantLapin** effectuait un appel au générateur aléatoire Mersenne Twister à chaque exécution, soit plusieurs millions d'appels par mois.

Pour remédier à ce problème, j'ai modifié l'approche : au lieu de générer un nombre aléatoire à chaque appel, je pré-génère un tableau de taille **tailleTabMersenneTwister** contenant tous les nombres aléatoires nécessaires au début de l'expérience. Ces valeurs sont ensuite réutilisées au fil de l'exécution.

Cette optimisation a permis de réduire le temps total d'exécution de 30 secondes à 8 secondes, dont seulement 0,12 seconde est désormais consacrée à la génération des nombres aléatoires.

3.6 Reproductibilité

Pour garantir la reproductibilité des simulations, j'ai pris soin de sauvegarder la seed utilisée pour le générateur aléatoire dans un fichier `log.txt`.

En enregistrant la seed dans le fichier de log, il est possible de reproduire exactement la même simulation ultérieurement, y compris les tirages aléatoires de chaque lapin. Cette approche permet de vérifier les résultats et de comparer différentes versions de la simulation avec exactement les mêmes conditions initiales.

Le fichier de log contient à la fois les valeurs simulées (population mois par mois) et la seed utilisée, assurant une traçabilité complète de l'expérience.

3.7 Résultats d'une simulation complexe

La simulation complexe, prenant en compte l'âge, la mortalité, le nombre de petits par portée et la maturité sexuelle, produit une évolution de population plus réaliste que le modèle Fibonacci. Les variations aléatoires dues aux tirages individuels sont visibles, mais la population suit globalement une croissance exponentielle.

3.7.1 Graphe en échelle normale

Évolution de la population de lapins sur 80 mois selon la simulation complexe (échelle linéaire).

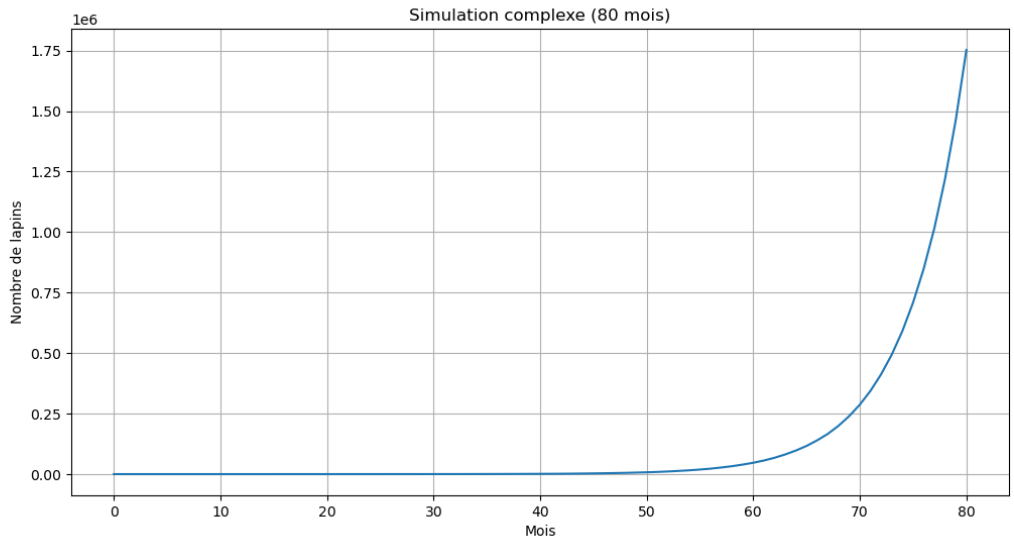

FIGURE 3 – Évolution de la population de lapins sur 80 mois selon la simulation complexe.

3.7.2 Graphe en échelle logarithmique

Évolution de la population de lapins sur 80 mois selon la simulation complexe (échelle logarithmique). L'axe log permet de mieux visualiser la croissance exponentielle tout en conservant les fluctuations aléatoires.

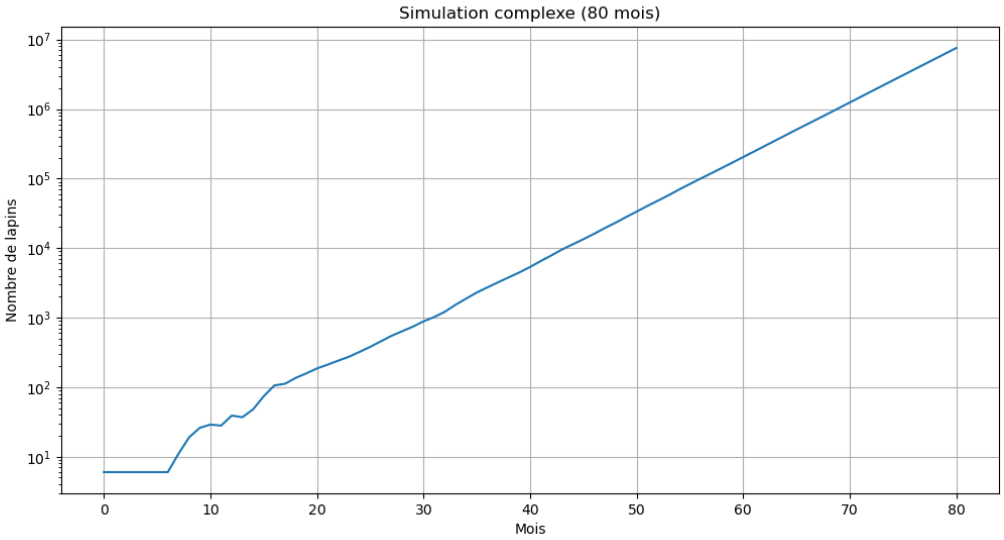

FIGURE 4 – Évolution de la population de lapins sur 80 mois selon la simulation complexe avec l'axe des ordonnées en log.

3.7.3 Comparaison entre les deux modèles

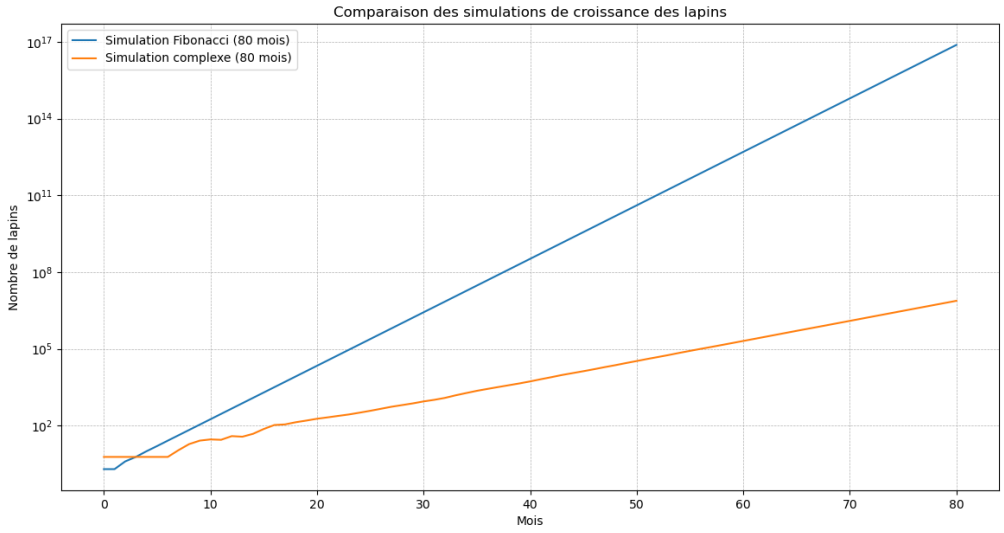

FIGURE 5 – Évolution de la population de lapins sur 80 mois selon la simulation complexe avec l'axe des ordonnées en log.

Comparé au modèle de Fibonacci, le modèle complexe fournit des résultats beaucoup plus réalistes. La croissance y est plus modérée et présente des fluctuations importantes dues aux effets aléatoires, contrairement à la croissance exponentielle et idéalisée du modèle de Fibonacci.

Remarque : la seed utilisée pour cette simulation était `[291, 564, 837, 1110]` pour le MersenneTwister, assurant la reproductibilité des résultats.

4 Analyse des résultats de la simulation de population de lapins

4.1 Résumé des résultats

La simulation sur 80 mois a été réalisée selon deux modèles :

— **Modèle de Fibonacci** : la population évolue selon la suite de Fibonacci classique multipliée par deux (nombre de couples de lapins). La croissance est exponentielle, avec les valeurs suivantes pour les premiers mois :

$$2, 2, 4, 6, 10, 16, 26, 42, 68, 110, 178, \ldots$$

La population atteint plus de 7.5×10^{13} lapins au 80^{e} mois.

— **Modèle complexe** : ce modèle inclut des facteurs réalistes tels que l'âge de maturité, le sexe des lapins, la probabilité de reproduction, et le risque de mortalité mensuelle et annuelle. La simulation est stochastique et a été répétée plusieurs fois pour obtenir des statistiques. Les résultats obtenus sont :
  — Médiane : 2 959 528 lapins
  — Min : 0 lapin
  — Max : 10 164 249 lapins
  — Moyenne : 3 170 858.7 lapins
  — Variance : 7.393×10^{12}
  — Écart-type : 2 719 029.98
  — Coefficient de variation : 85.75 %
  — Intervalle de confiance à 95% : [2 157 159, 4 184 558]
  — Pourcentage d'extinction : 16.67%

4.2 Analyse comparative

— **Croissance** : le modèle de Fibonacci prédit une croissance exponentielle et irréaliste de la population, alors que le modèle complexe montre une population beaucoup plus modérée et soumise aux fluctuations aléatoires. La différence principale vient des contraintes biologiques et du hasard intégré dans le modèle complexe.

— **Variabilité** : le coefficient de variation élevé (85.75%) dans le modèle complexe indique une grande dispersion autour de la moyenne, ce qui reflète l'effet du hasard sur la reproduction et la mortalité. La médiane étant légèrement inférieure à la moyenne suggère que quelques simulations avec une population très élevée ont tiré la moyenne vers le haut.

— **Risque d'extinction** : le fait que 16.67% des simulations aient abouti à une extinction totale des lapins montre que le modèle complexe capture la possibilité de défaillance de la population, contrairement au modèle de Fibonacci où la population ne peut jamais décroître.

— **Intervalle de confiance** : l'intervalle de confiance à 95% pour la population finale montre que, dans la majorité des cas, la population restera comprise entre environ 2,1 millions et 4,2 millions de lapins, bien loin des valeurs astronomiques prédites par Fibonacci.

4.3 Conclusion

Le modèle complexe fournit une représentation beaucoup plus réaliste de l'évolution d'une population de lapins, intégrant les effets du hasard, de la mortalité et des contraintes biologiques. En comparaison, le modèle de Fibonacci est purement mathématique et surestime largement la croissance de la population. Ces résultats mettent en évidence l'importance de prendre en compte les facteurs stochastiques et biologiques pour des simulations fiables de populations.

Annexes

Description des fichiers sources et de sortie

makefile

Ce fichier permet de compiler les fichiers, éxécuter le main, puis de générer les images avec le script python juste en faisant **make**.

main.c

Ce fichier contient la fonction principale du programme. Il initialise les générateurs aléatoires (`rand()` et Mersenne Twister), lance les simulations selon le modèle de Fibonacci et le modèle complexe, mesure le temps d'exécution et écrit les résultats dans un fichier `log.txt`.

randomGenerator.c

Implémente les fonctions de génération aléatoire utilisées dans la simulation. On y retrouve notamment :

- une génération uniforme entre deux bornes entières ;
- une génération suivant une distribution gaussienne (méthode de Box–Muller) ;
- des appels au générateur Mersenne Twister.

randomGenerator.h

Déclare les fonctions présentes dans `randomGenerator.c`. Ce fichier permet de séparer les définitions des fonctions de leur utilisation dans le reste du projet.

TP4.c

Contient l'ensemble des fonctions de simulation des populations de lapins. Il inclut notamment :

- la simulation selon la suite de Fibonacci ;
- la simulation complexe prenant en compte mortalité, reproduction et vieillissement ;
- les fonctions d'affichage des résultats dans la console.

TP4.h

Fichier d'en-tête associé à `TP4.c`, déclarant toutes les fonctions et structures nécessaires à la simulation des lapins.

log.txt

Fichier texte généré automatiquement à chaque exécution du programme. Il contient :

- les valeurs générées pour la suite de Fibonacci et la simulation complexe ;
- la `seed` utilisée pour les générateurs aléatoires, permettant de rejouer exactement une simulation.

graphe.py

Script Python servant à la génération des graphiques à partir des données du `log.txt`. Il lit les résultats, trace les courbes de croissance des populations et permet d'afficher les comparaisons entre modèles.

Images PNG (résultats de simulation)

Les fichiers `.png` représentent les graphiques obtenus à partir du script Python. Ils illustrent respectivement la croissance selon le modèle de Fibonacci, la simulation complexe et la comparaison des deux approches.

Références

[1] Wikipédia, *Suite de Fibonacci*, https://fr.wikipedia.org/wiki/Suite_de_Fibonacci

[2] Wikipédia, *Loi normale*, https://fr.wikipedia.org/wiki/Loi_normale

[3] Wikipédia, *Mélange de Fisher–Yates*, https://fr.wikipedia.org/wiki/M%C3%A9lange_de_Fisher%E2%80%93Yates

[4] Wikipédia, *Mersenne Twister*, https://fr.wikipedia.org/wiki/Mersenne_Twister

[5] Microsoft Learn, *srand*, https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/srand?view=msvc-170