Rapport TP4 - Projet Informatique

LOURENÇO Cynthia et YMERAJ Eldis

04/04/2022

Figure 1 – Image d'accueil de lapin

Table des matières

1 Introduction

Dans ce rapport, nous allons voir la croissance d'une population.

Mais qu'est ce qu'une modélisation ?

La modélisation informatique des données est en réalité un processus de description de la structure, des associations, des relations et des impératifs liés à des datas disponibles. Elle permet de fixer des normes, tout en codant des modèles de gestion des données dans une organisation.

Ici, nous nous intéressons à la modélisation d'une population.

Donc, voyons ce qu'est le dynamisme d'une population ?

Le dynamisme des populations est une branche de l'écologie qui s'intéresse à la fluctuation dans le temps du nombre d'individus au sein d'une population d'êtres vivants. Elle a également pour but de comprendre les influences environnementales sur les effectifs des populations. La structuration de la population par âge, taux de mortalités, l'environnement et les processus qui influent sur ces changements font également partie de son champ d'étude.

Quelles sont les possibilités de résultat face à cette modélisation ?

Les modèles de croissance géométrique et exponentielle supposent un taux de croissance proportionnel au nombre d'individus. Leur dynamique est assez simple et peu réaliste : la population croît indéfiniment, cependant leurs prédictions peuvent se révéler correctes sur des temps courts. Ils servent de base à des modèles plus sophistiqués comme les modèles matriciels de populations pour étudier la dynamique des populations structurées.

Une population présentant une dynamique stable ou cyclique peut évoluer vers une dynamique chaotique. Cette évolution pourrait être due à des changements des conditions environnementales, à des phénomènes de prédation ou de dispersion. Cette transition a été décrite mathématiquement par plusieurs modèles qui en fonction des paramètres qu'ils étudient (dispersion, prédation, conditions environnementales) mènent à des dynamiques chaotiques (i.e. Diagramme de Feigenbaum).

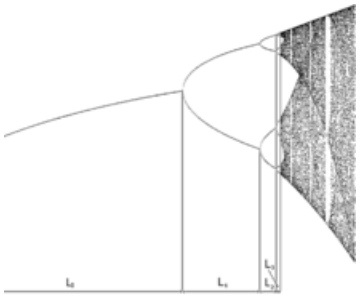

Figure 2 – Diagramme de Feigenbaum

Nous allons modéliser plusieurs type de croissance d'une population.

D'abord, nous verrons la suite de fibonacci qui peut simuler la croissance de lapin de manière peu réaliste puis nous verrons une autre manière de simuler la croissance d'une population de lapin avec des contraintes définis comme le taux de mortalité par exemple.

2 Le concept de Fibonacci

Dans cette première partie de projet on va étudier une simulation rapide de la croissance d'une population de lapins. Premièrement on a un couple de lapins et on a considéré qu'un couple de jeunes lapins va devenir adultes dans un mois et quand la femelle est enceinte, ça prend un mois de nous donner deux bébés, d'où ces bébés dans un mois seront adultes. Dans la première partie on a supposé que les lapins ne meurent pas et on a le diagramme dessous :

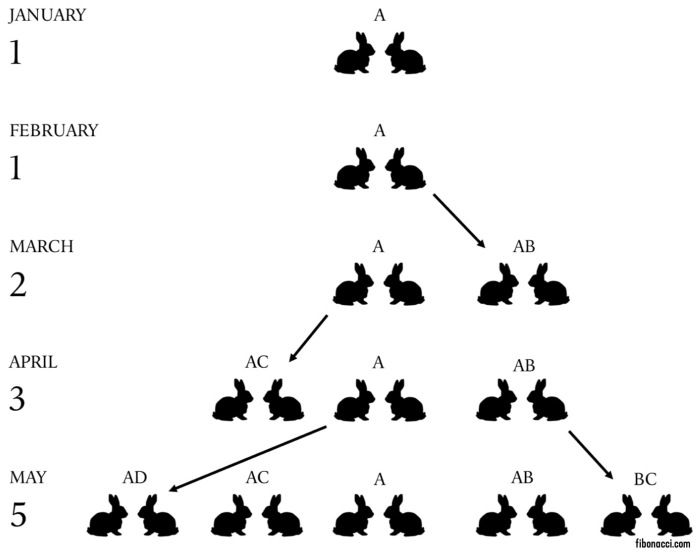

Figure 3 – observation faite par Léonard de Pizza

La simulation a un pas de temps discret d'un mois. À la date 1 Janvier on a un couple de lapins bébé et à la date 1 Février, après un mois le couple est maturé pour donner des portées. À la date 1 Mars on a un nombre de couples par mois 'N' qui se calcule : nombre de couples par mois 'N-1' en ajoutant le nombre de couples par mois 'N-2' et à l'étape 4 le nombre actuel de couples est affiché.

On peut calculer tout ça avec Fibbonacci d'où : U(N)=U(N-1)+U(N-2). La fonction 'fib' :

```
1  public static long fib(int n){
2      if(n <= 1)
3      {
4          return n;
5      }
6      else
7      {
8          return fib(n-1) + fib(n-2);
9      }
10 }
```

Grâce à la fonction 'fib' on calcule le nombre de couples comme dans la Fig.1.

En utilisant cette fonction, on peut voir cette suite apparaître :
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

On peut donc voir que cette fonction a un taux de croissance exponentiel du fait que l'on calcule chaque terme avec la somme des deux termes précédents.

Mais si cette suite est aussi célèbre aujourd'hui, c'est parce qu'elle apparaît dans plusieurs aspects de la vie comme l'art, l'architecture et même la nature elle-même. On peut la voir sur le tableau de la Joconde de Léonard de Vinci, une main humaine, un coquillage en spiral ou encore dans un chou romanesco. Elle apparaît comme une spirale.

Figure 4 – Chou romanesco et ses spirales

On peut calculer cette spirale grâce au nombre d'or issu de la suite de Fibonacci. Sa valeur exacte est de (1+5)/2, ayant comme dix premières décimales 1,6180339887... Ce rapport, considéré comme la clé de l'harmonie universelle, se décline et se transpose par des formes géométriques telles que le rectangle, le pentagone et le triangle.

Ici, on étudie la croissance d'une population de lapin et on peut voir que la suite Fibonacci n'est pas très représentative de cela car elle ne prend pas en compte l'infertilité, les prédateurs ou encore du nombre de portées de la lapine dans l'année. Cette suite n'étudie qu'un seul couple de lapin.

Nous allons maintenant voir une version de croissance d'une population de lapin plus réaliste que celle de Fibonacci.

3 Rabbit simulation

3.1 Description

Dans cette section, nous allons implémenter une simulation de croissance de lapin avec différents paramètres qui détermine une modélisation plus réaliste :

1. un genre aléatoire entre mâle et femelle à chaque naissance ;
2. un âge de maturité qui permet de pouvoir enfanter ;
3. le taux de survie en fonction de son âge ;
4. la fertilité de chaque lapin ;
5. le nombre de portées par an d'une lapine ;
6. le nombre de lapereau né dans chaque portée.

Les données de chaque paramètres sont :

Genre

On choisit aléatoirement à la naissance entre 2 genres équiprobablement. Ici, un lapereau peut être un mâle à 50% de chance ou une femelle aussi à 50% de chance.

Maturité

On choisit un âge où le lapereau devient un lapin. Cet âge se compte en mois. Chaque lapereau a :
- 5% de chance que sa maturité soit de 4 mois ;
- 15% de chance que sa maturité soit de 5 mois ;
- 35% de chance que sa maturité soit de 6 mois ;
- 25% de chance que sa maturité soit de 7 mois ;
- 15% de chance que sa maturité soit de 8 mois ;
- et 5% de chance que sa maturité soit de 8 mois.

Taux de survie

- Pour les lapereaux, c'est-à-dire les lapins qui n'ont pas encore atteint l'âge de leur maturité, ont un taux de survie de 50% pour l'année.
- Pour les lapins qui ont un âge entre leur maturité et 7 ans ont un taux de survie de 75%
- Pour les lapins de 8 ans, ils ont un taux de survie de 60%
- Chaque année, le taux de survie décroît de 15% qui passe de 8 ans jusqu'à 12 ans où il meurt automatiquement à cet âge-là ;
- Chaque lapin ont 7% de chance d'être manger par un renard chaque mois quelque soit son âge (la population de renard croît comme la population de lapin donc plus il y a de lapin, plus il y a de renard pour manger des lapins).
- A chaque accouchement, une lapine a 15% de chance de mourir.

Fertilité

Chaque lapin et lapine mature ont 10% de chance d'être infertile durant chaque mois.

Nombre de Portée

Chaque lapine ont :
- 1% de chance d'avoir 2 portées dans l'année ;
- 4% de chance d'avoir 3 portées dans l'année ;
- 10% de chance d'avoir 4 portées dans l'année ;
- 17% de chance d'avoir 5 portées dans l'année ;
- 35% de chance d'avoir 6 portées dans l'année ;
- 19% de chance d'avoir 7 portées dans l'année ;
- 9% de chance d'avoir 8 portées dans l'année ;
- 3% de chance d'avoir 9 portées dans l'année ;
- 1% de chance d'avoir 10 portées dans l'année ;

Nombre de lapereau pour chaque portée

A chaque portée, il y a :
- 5% de chance d'avoir 2 lapereaux ;
- 15% de chance d'avoir 3 lapereaux ;
- 30% de chance d'avoir 4 lapereaux ;
- 30% de chance d'avoir 5 lapereaux ;
- 15% de chance d'avoir 6 lapereaux ;
- 5% de chance d'avoir 7 lapereaux.

Perspective

De là on peut imaginer que les lapins seront exponentielles même si il y a plus de paramètres qui rentre en compte que pour Fibonacci. Car, il y a beaucoup de portées dans l'année donc beaucoup de lapins qui naissent et le taux de survie tue moins de lapins qu'il en naissent dans une année pour une initialisation de 4 lapins par exemple.

3.2 Makoto Matsumoto pseudo random-number

Dans le code Matsumoto, on peut générer plusieurs type de nombres pseudo-aléatoires.

Il existe :
- genrandint31() qui génère des nombres entiers ;
- genrandreal1() qui génère des nombres dans [0,1] ;
- genrandreal2() qui génère des nombres dans [0,1[;
- genrandreal3() qui génère des nombres dans [0,1[;
- genrandres53() qui génère des nombres dans [0,1[avec 53-bit resolution.

Ces fonctions peuvent êre très intéressante avec une bonne initialisation dans le *main*.

Dans la suite de mon code, on utilisera que genrandreal1() et genrandreal2().

On les utilise pour pouvoir générer des nombres pseudo-aléatoires. Ils sont issus d'un algorithme qui génère une séquence de nombres présentant certaines propriétés du hasard. Par exemple, les nombres sont supposés être suffisamment indépendants les uns des autres, et il est potentiellement difficile de repérer des groupes de nombres qui suivent une certaine règle.

On utilisera ces fonctions notamment pour la fertilités, le taux de survie, le nombre de portée d'une lapine par an, le nombre de lapereau par portée ou encore pou généré leurs genres.

Ces fonctions sont dans le fichier *Matsomoto.c* qui est répertorier dans le ficher *Matsumoto.h*.

3.3 Choix d'achitecture

Pour pouvoir simuler une population de lapin, on a décider d'implémenter une structure de liste doublement chaînée ListeLapin qui possède le nombre d'éléments.
Cette liste prend en paramètre un pointeur sur une structure Lapin qui se trouve à la tête de la liste ListeLapin ainsi qu'un pointeur qui pointe sur la queue de la liste ListeLapin.

Voici le code associé à la structure ListeLapin :

```
1  struct ListeLapin
2  {
3      int nb_elements;
4      Lapin * head;
5      Lapin * tail;
6  };
```

La liste ListeLapin comprend une liste de Lapin possèdant plusieurs paramètres comme :

- next : un pointeur sur l'élément suivant de la liste
- prev : un pointeur sur l'élément précèdent de la liste
- un entier sexe : 0 pour mâle et 1 pour femelle
- un entier age : l'âge courant du lapin en mois si l'âge du lapin est -1 alors il est mort
- un entier maturite : l'âge à laquel le lapin devient mature
- un entier fertil : si le lapin est fertile 1 ou pas 0
- un entier nbPortee : le nombre de portée qu'une lapine a dans l'année
- un entier nbMoisPortee : le décompte dans laquelle la lapine ne peut pas avoir de lapereau car la lapine accouche les 1er mois (cad 5 nombre de portée alors la lapine accouche les 5 1er mois) puis pendant le reste de l'année (12-nbPortee) on attend qu'elle soit fécondable

Voici le code associé :

```
1  struct Lapin
2  {
3      Lapin * next;
4      Lapin * prev;
5      int sexe;
6      int age;
7      int maturite;
8      int fertil;
9      int nbPortee;
10     int nbMoisPortee;
11 };
```

3.4 Implémentation des fonctions

3.4.1 generateurMaturite() :

- Génère un chiffre qui représente la maturité choisit aléatoirement.
- Valeur de retour de type int.
- Entrée : rien.
- En sortie : un chiffre alétoire entre 5 et 8 avec genrandreal1().
  - si le random est en dessous de 0.05 alors la valeur de retour est 4.
  - si le random est en dessous de 0.2 alors la valeur de retour est 5.
  - si le random est en dessous de 0.2 alors la valeur de retour est 6.

- si le random est en dessous de 0.2 alors la valeur de retour est 7.
- si le random est en dessous de 0.2 alors la valeur de retour est 8.
- sinon la valeur de retour est 9.

Signature de fonction :

```
int generateurMaturite()
```

3.4.2 generateurMaleFemelle() :

- Génère un chiffre qui représente le sexe choisit aléatoirement.
- Valeur de retour de type int.
- Entrée : rien.
- En sortie : un chiffre aléatoire entre 0 (mâle) et 1 (femelle) avec genrandreal1().
  - Si il est en dessous de 0.5 alors la valeur de retour est 0.
  - Sinon 1.

Signature de fonction :

```
int generateurMaleFemelle()
```

3.4.3 lapereau(...) :

- Lapereau : un Lapin plus le nouveau Lapin initialisée
- Valeur de retour : ListeLapin.
- Entrée : la liste de lapin.
- En sortie :
  - Crée un nouveau élément dans la liste Lapin initilalisé comme un bébé lapin.
  - on génère le sexe avec generateurMaleFemelle().
  - on génère l'âge de la maturité avec generateurMaturite().
  - le reste des paramètres de Lapin est initialisé à 0 ou NULL pour les pointeurs, ensuite on regarde si la liste est vide ou pas pour reinitilaliser nbelements, la tail et le head de la liste.

Signature de fonction :

```
ListeLapin * lapereau( ListeLapin* l )
```

3.4.4 mort(...) :

- mort : tue un lapin.
- Valeur de retour : Liste.
- Entrée : Un lapin.
- En sortie : un lapin avec toutes ses paramètres à 0 sauf pour l'âge qui est à -1 qui signifie la mort ici

Signature de fonction :

```
Lapin * mort( Lapin * E )
```

3.4.5 fertilite(...) :

- fertilite : Fertilité initilalisé dans la liste.
- Valeur de retour : Lapin.
- Entrée : Un lapin.
- En sortie : initialise aléatoirement un lapin matures leurs fertilités qu'il soit mâle ou femelle avec une probabilité de 10 pourcents pour l'infertilité.
  - si le lapin a atteint ou dépassé son âge de maturité.
  - on prend un nombre pseudo-aléatoire du code de makoto.
  - si ce nombre est inférieur à 0.1 alors il est infertile 0 sinon il est fertile.

Signature de fonction :

```
1 Lapin * fertilite( Lapin * E )
```

3.4.6 auMoinsUnMale(...) :

- Est un fonction qui vérifie s'il y a au moins un mâle fertile, vivant et mature.
- Valeur de retour : booléen.
- Entrée : La listeLapin
- En sortie : Premièrement notre fonction va vérifier s'il existe dans laliste un mâle fertile, s'il est fertile alors il est maturé et vivant dès qu'on en trouve un on renvoie true, sinon false.

Signature de fonction :

```
1 bool auMoinsUnMale( ListeLapin * l )
```

3.4.7 nbLapereauParPortee() :

- Cette fonction va nous générer un chiffre qui représente le nombre de bébés par portée choisit aléatoirement entre 2 et 7.
- Valeur de retour : entier.
- Entrée : Rien.
- En sortie : Nous allons avoir un chiffre alétoire entre 5 et 8 avec l'utilisation de fonction genrandreal1(), avec les cas suivantes :
  - Si le random est en dessous de 0.05 alors la valeur de retour est 2.
  - Si le random est en dessous de 0.2 alors la valeur de retour est 3.
  - Si le random est en dessous de 0.5 alors la valeur de retour est 4.
  - Si le random est en dessous de 0.8 alors la valeur de retour est 5.
  - Si le random est en dessous de 0.95 alors la valeur de retour est 6.
  - Sinon la valeur de retour est 7.

Signature de fonction :

```
1 int nbLapereauParPortee()
```

3.4.8 nbPortee(...) :

- Est un fonction qui détermine le nombre de portée d'une lapine par an.

— Valeur de retour : Lapin.
— Entrée : Un lapin et un tableau qui contient toutes les probabilités pour le nombre de portées.
— En sortie nous allons avoir un Lapin modifié avec le nombre de portées que chaque femelle a donné si celles-ci sont fertiles, en plus son compteur de nbMoisPortee est 0 et qu'il existe au moins un mâle qui puis la fécondé.
  — Si toutes les conditions sont remplis alors, on choisit psr à manière aléatoire et on fait le comparaison avec notre tableau
  — Si le psr est inférieur que l'une des valeurs de tableau alors, on incrémente le nbPortee de la lapine de l'index du tableau+2, en plus on incrémente nbMoisPortee de 12 pour dire que pendant 12 mois.
  — À la fin la lapine ne pourra pas avoir des nouvelles portées.

Signature de fonction :

```
1 Lapin * nbPortee( Lapin * E, double tabProbaPortee[9], bool auMoinsUnMale )
```

3.4.9 naissance(...) :

— Cette fonction permet de créer des nouveaux lapereaux en fonction d'une portée.
— Valeur de retour : entier.
— Entrée : Un lapin.
— En sortie, si le nombre de portées est supérieur à 0 alors, nbLapereauParPortee() lapereau est créé et le nbPortee est décrémenté de 1 en utilisant un nombre random.
  — Si le nombre random est inférieur à 0.15 alors la lapine qui vient de créer une portée, meurt.

Signature de fonction :

```
1 int naissance( Lapin * E )
```

3.4.10 ageMois(...) :

— Cette fonction permet de créer des nouveaux lapereaux en fonction d'une portée.
— Valeur de retour : Lapin.
— Entrée : Un lapin.
— En sortie nous allons avoir un Lapin d'où son âge est incrémenté de 1 pour les lapins vivants et pour les lapines qui ont le nbMoisPortee strictement positif alors, nbMoisPortee est décrémenté de 1.

Signature de fonction :

```
1 Lapin * ageMois( Lapin * E )
```

3.4.11 tauxDeSurvie(...)

— Cette fonction permet de calculer le taux de survie en fonction de son âge pour chaque lapin.
— Valeur de retour : Lapin.
— Entrée : Un Lapin.
— En sortie, si un lapin est mort nous allons le modifier, on va vérifier si le lapin n'est pas déjà mort avant de calculer son taux de survie et on initialise un renard qui peut manger des lapins selon un

nombre pseudo-aléatoire, et un nombre pseudo-aléatoire pour calculer le taux aléatoire ,pour chaque année de la vie d'un il a un certain taux de survie. Le taux de survie est calculé en mois. On a les conditions suivantes :

- S'il n'est pas mature et son taux de survie est inférieure à tauxEnmois(0.5) alors il meurt.
- Si son âge est entre maturité et 8 ans (96 mois) et qu'il a un taux de survie inférieur à tauxEnMois(0.75) alors il meurt.
- S'il a 8 ans(96 mois) et son taux de survie est inférieur à tauxEnMois(0.6) alors il meurt.
- S'il a 9 ans(108 mois) et son taux de survie est inférieur à tauxEnMois(0.45) alors il meurt.
- S'il a 10 ans(120 mois) et son taux de survie est inférieur à tauxEnMois(0.3) alors il meurt.
- S'il a 11 ans(132 mois) et son taux de survie est inférieur à tauxEnMois(0.15) alors il meurt.
- S'il a 12 ans(96 mois) alors, il meurt.
- Si le renard est inférieur à 0.15 alors, le renard a mangé le lapin.

Signature de fonction :

```
1 Lapin * tauxDeSurvie( Lapin * E )
```

3.4.12 Main

Tout d'abord dans le main, on initialise le code de Makoto ensuite on initialise toutes les variables comme la ListeLapin, le nombre de mois *nbMois_t*, le nombre de Lapin initial *n_t* ainsi que le tableau des probabilitées pour le nombre de portée *tabProbaPortee*.

On commence par ajouter le nombre *n_t* de lapereau voulu.

Et puis, dans une première boucle qui prend comme itérations le nombre de mois, on initialise le début de ListeLapin *E* puis on vérifie si il y a au moins un mâle fertile *maleFertil* ensuite on initialise le nombre de lapin en vie *lapinEnVie* et le nombre de naisance *naissance* pour chaque mois.

Dans cette boucle, on prend une autre boucle qui on parcourt toute la liste de Lapin, on vérifie si chaque lapin est fertile *fertilite(E)*, puis on voit si il peut avoir une portée *nbPortee(E, tabProbaPortee)*, ensuite on incrémente la naissance si il y en a *naissance(E)*, on calcule ensuite le taux de suivie *tauxDeSurvie(E)*. A partir de là, toujours dans les boucles incrémente le nombre de lapin en vie et enfin on incrémente l'âge des lapins de 1.

Enfin, en sortant de la boucle de la liste de lapin, on crée de nouveaux lapins grâce aux nombres de lapins issus de *naissance* et on écrit les résultats dans le terminal.

Fianlement en sortant de la dernière boucle, on libère la liste.

```c
int main(){

    // Initialisation de Matsumoto
    unsigned long init[4]={0x123, 0x234, 0x345, 0x456}, length=4;
    init_by_array(init, length);

    // Initialisation liste et constante
    ListeLapin *l                   = new_liste();
    double      tabProbaPortee[9]   = {0.01, 0.05, 0.16, 0.33, 0.68, 0.87, 0.96, 0.99, 1};
    int         n_t                 = 0,
                duree_t             = 0;

    printf("Combien de lapins voulez-vous pour commencer la simulation ?\n");
    scanf("%d", &n_t);

    printf("Pendant combien de temps voulez-vous que la simulation dure en mois ? (conseille
     pas plus de 80 mois)\n");
    scanf("%d", &duree_t);

    // Création de n lapins comme initialisation
    for( int i = 0; i < n_t; i++ )
    {
        l = lapereau(l);
    }

    // Simulation pendant duree_t mois pour toute la liste de lapin
    for( int nbMois = 0; nbMois < duree_t; nbMois++ )
    {
        Lapin * E           = l -> head;
        bool    maleFertil  = auMoinsUnMale(l);
        int     lapinEnVie  = 0,
                nouveauBebe = 0;

        for(int i = 0; i < l -> nb_elements; i++)
        {
            E           = fertilite( E );
            E           = nbPortee( E, tabProbaPortee, maleFertil);
            nouveauBebe += naissance( E );
            E           = tauxDeSurvie( E );

```

```
40              if( E -> age != -1 )
41              {
42                  lapinEnVie++;
43              }
44
45              E           = ageMois( E );
46              E           = E -> next;
47          }
48
49          for( int j = 0; j < nouveauBebe; j++)
50          {
51              l = lapereau( l );
52              lapinEnVie++;
53          }
54
55          printf("mois: %d\nLapins en vie: %d\nLapins morts: %d\nLapins durant l'experience: %
        d\n\n", nbMois, lapinEnVie, l -> nb_elements - lapinEnVie, l -> nb_elements);
56      }
57
58      // Fin de la simulation, libération de la liste
59      freeListe( l );
60
61      return 0;
62  }
```

3.5 Résultat de la modélisation

En initialisant 5 lapereaux et en faisant une simulation sur 9 ans c'est-à-dire sur 108 mois, on obtient :

mois : 108
Lapins en vie : 10775128
Lapins morts : 5465494
Lapins durant l'experience : 16240622

Sur un graphiques en fonction du nombre de lapin en vie et le temps en années, on a :

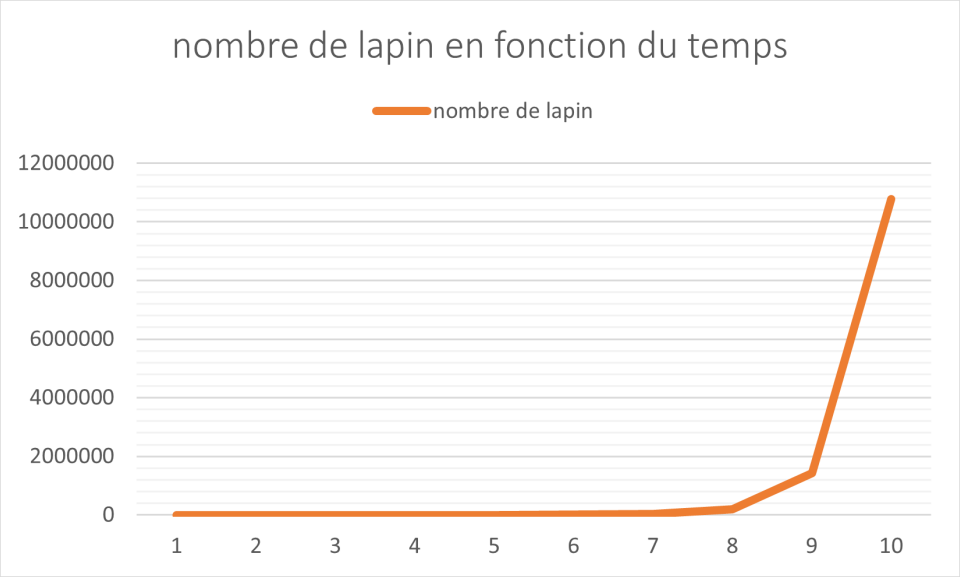

Figure 5 – Courbe du nombre de lapin en fonction du temps

On voit bien que la croissance des lapins est exponetielle et continue comme pour Fibonacci.

On peut maintenant essayer de voir plusieurs expérimentations pour en faire une moyenne, en changeant l'initialisation du code de Makoto, on obtient ces valeurs pour le nombre de lapin en vie :

nombre de lapins	10775128	11282153	44341875	42880288	31514316	30955255	31684653
moyenne	29061952						

En essayant de calculer l'erreur relative avec cette formule :

$$ErreurRelative = \frac{meanA - A}{A} \tag{1}$$

On trouve ici, une erreur relative de -0.6. Ce qui est beaucoup.

On pense que calculer l'intervalle de confiance ne sert à rien au vu de la grande différence entre les nombres de lapin en vie lorsque l'initialisation de Makoto est différente.

4 Conclusion

Dans ce projet nous sommes arrivés de simuler la croissance d'une population de lapins avec des contraintes définis. Nous avons étudié comment une population de lapins ça change chaque mois et chaque année.

Nous avons pris en compte chaque risque éventuel qu'un lapin peut rencontrer au cours de sa vie et on a établi des probabilités appropriées pour que notre étude sera très réaliste. Pour les prochaines générations de lapins (les lapereaux) , nous avons bien étudie le nombre de portées qu'un lapine peut donner par an en donnant son probabilité appropriée.

Nous avons fait tout ça à l'aide des fonctions de Makoto Matsumoto qu'ils ont nous aidé pour générer des nombres pseudo-aléatoires que notre étude sera très réelle.

Grâce à notre étude, on conclure qu'un petit nombre de lapins (mâle et femelles) peut générer un très grand nombre de cette population dans une courte période en prennent en compte toutes les risques de la vie.

Table des figures