Dynamique des populations : étude du risque de surpopulation
Annexes du TIPE

DUBARD Loïc,
MONNIER Clément,
ALCARAZ Angelo

22 juin 2018

Table des matières

I Démonstrations 5

II Scripts Python 13

1 Interface graphique 15

2 Modèle de Leslie 23

3 Modèles discret et continu 35

4 Méthode de la puissance 41

5 Rayon spectral de Leslie 43

6 Mc Kendrick/Von Foerster 47

7 Transition démographique 51

8 Paramètres gaussienne 55

Première partie

Démonstrations

Modèle de Leslie & équation de McKendrick/Von Foerster (application à la démographie)

1 Modèle de Leslie et théorème de Perron-Frobenius

Introduction. On va définir ce qu'est une matrice de Leslie et montrer qu'on peut lui appliquer les hypothèses du théorème de Perron-Frobenius. Nous verrons ensuite les applications d'un tel résultat à la démographie.

Dans toute la suite, on se donne $n \in \mathbb{N} - \{0\}$. Pour X un ensemble, on notera $\mathcal{M}_n(X)$ l'ensemble des matrices carrées de taille n à coefficents dans X. On note $\mathbb{R}$ (resp. $\mathbb{R}_+$) l'ensemble des réels (resp. des réels positifs) et $\mathbb{R}^n$ (resp. $\mathbb{R}_+^n$) l'ensemble des vecteurs à n composantes réelles (resp. réelles positives).

1.1 Définitions

Définition 1. *Soit M une matrice de $\mathcal{M}_n(\mathbb{R})$, on dit que c'est une matrice de Leslie si elle est de la forme suivante, avec $(f_1, ..., f_m) \in \mathbb{R}_+^m$ et $(s_1, ..., s_{m-1}) \in \mathbb{R}_+^{m-1}$:*

$$\begin{pmatrix} f_1 & f_2 & \cdots & f_{m-1} & f_m \\ s_1 & 0 & \cdots & 0 & 0 \\ 0 & s_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & s_{m-1} & 0 \end{pmatrix}$$

Définition 2. *Soit M une matrice de $\mathcal{M}_n(\mathbb{R}_+)$, on dit qu'elle est primitive si elle est à coefficients positifs et qu'il existe une puissance k telle que M^k est à coefficients strictement positifs*

Définition 3. *Soit M une matrice de $\mathcal{M}_n(\mathbb{R}_+)$, on dit qu'elle est irréductible si quelquesoit $k \in \mathbb{N} - \{0\}$, il n'existe pas de permutation des vecteurs de la base canonique pour laquelle la matrice M^k serait triangulaire par blocs i.e de la forme suivante:*

$$M^k \sim \begin{pmatrix} X & Y \\ 0 & Z \end{pmatrix}$$

Lemme 4. *Soit M une matrice de $\mathcal{M}_n(\mathbb{R}_+)$, si M est primitive alors M est irréductible*

Preuve.

Si M n'est pas irréductible : $\exists k \in \mathbb{N} - \{0\}, M^k \sim \begin{pmatrix} X & Y \\ 0 & Z \end{pmatrix}$

Si $k = 1$ alors aucune puissance de M n'est à coefficients strictement positifs.

Si $k > 1$: Soit $j \in \{1, k-1\}$ alors forcément M^j n'est pas à coefficients strictement positifs sinon M^k le serait. Soit $n \in \mathbb{N}$, on considère la division euclidienne de n par k : $n = p\,k + j$. Donc finalement, $M^n = M^{pk}\,M^j$.

Or, M^j et M^{pk} ne sont pas à coefficients strictement positifs donc leur produit non plus (considérer un produit par blocs car dans les deux matrices apparaît un bloc nul au même endroit par permutation des vecteurs de la base canonique).

Ainsi, aucune puissance de M n'est à coefficients strictement positifs donc M est primitive.

Lemme 5. *Soit $M=(m_{i,j})_{1\leqslant i,j\leqslant n}$ une matrice de $\mathcal{M}_n(\mathbb{R}_+)$, si M est irréductible alors la matrice $(I_n+M)^{n-1}$ est à coefficients strictement positifs, où I_n représente la matrice identité.*

Preuve.

Soit $X=(X_1,...,X_n)\in\mathbb{R}_+^n$ un vecteur non nul et $Y=(I_n+M)\,X=(Y_1,...,Y_n)$.

$\forall i\in\{1,...,n\}, Y_i=X_i+\sum_{k=1}^n m_{i,k}\,X_k \quad (1)$

Si X n'a que des composantes strictement positives alors Y aussi.

Si X possède p coordonnées strictement positives $(1\leqslant p<n)$ alors il est clair par (1) que Y aussi. On va prouver que Y en possède au moins une de plus.

On pose $K_p=\{k_1,...,k_p\}$ l'ensemble des indices des p coordonnées de X strictement positives. On suppose par l'absurde que si $j\notin K_p$ alors $Y_j=0$. Pour j un tel indice :

$$\sum_{k\in K_p} m_{j,k}X_k=0$$

Or, $k\in K_p\Rightarrow X_k>0$ donc $k\in K_p\Rightarrow m_{j,k}=0$. En considérant la permutation des vecteurs canoniques telle que : $(e_1,...,e_p,e_{p+1},...,e_n)\longrightarrow(e_k)_{k\in K_p}\cup(e_k)_{k\notin K_p}$ alors la matrice M est triangulaire par blocs dans cette base. Cela contredit l'irreductibilité de M donc Y possède un terme non nul de plus que X.

A fortiori, si $X\in\mathbb{R}_+^n$ est non nul alors $(I_n+M)^{n-1}X$ est à coefficients strictement positifs. Cela étant vrai en particulier pour la famille $(e_1,...,e_n)$ on obtient bien que $(I_n+M)^{n-1}$ est à coefficients strictement positifs.

Lemme 6. *Soit $M=(m_{i,j})_{1\leqslant i,j\leqslant n}$ une matrice de $\mathcal{M}_n(\mathbb{R}_+)$, si M est irréductible alors la fonction $r: X\in\mathbb{R}_+^n-\{0\}\longrightarrow\min_{X_i\neq 0}\frac{(MX)_i}{X_i}$ admet un maximum $\rho\geqslant 0$ sur $\mathbb{R}_+^n-\{0\}$.*

Preuve.

Posons $C=\{X\in\mathbb{R}_+^n,\sum_{1\leqslant i\leqslant n}(X)_i=1\}$, C est un compact car fermé et borné en dimension finie. De plus r est continue sur $\mathbb{R}_+^{\star n}$ comme minimum de n fonctions continues sur cet ensemble et si $X\in\mathbb{R}_+^{\star n}$ et si $t>0$ alors $r(t\,X)=r(X)$.

On pose maintenant $\tilde{C}=(I_n+M)^{n-1}(C)$ qui est un compact comme image continue du compact C et il est inclus dans $\mathbb{R}_+^{\star\,n}$ d'après le Lemme 5. Ainsi r est continue sur $\tilde{C}$ et y atteint un maximum noté ρ. Nous allons voir que ρ est aussi le maximum de r sur C.

Soit $X\in C$ et $Y=(I_n+M)^{n-1}\,X\in\tilde{C}$. Par définition de $r(X)$ on a pour tout i : $(M\,X)_i\geqslant r(X)\,X_i$ donc le vecteur $M\,X-r(X)\,X$ est a coordonnées positives. Or, M et $(I_n+M)^{n-1}$ commutent donc $(I_n+M)^{n-1}\,(M\,X-r(X)\,X)=A\,Y-r(X)\,Y$ et ce vecteur est à coordonnées positives par le Lemme 5. Donc pour tout i, $(A\,Y)_i\geqslant r(X)\,Y_i\Rightarrow r(Y)\geqslant r(X)$. Ainsi r est majorée par ρ sur C, et soit $Y\in\tilde{C}$ tel que $r(Y)=\rho$ alors en prenant $\tilde{Y}=\frac{Y}{\sum_{1\leqslant i\leqslant n}Y_i}\in C$, on a $r(\tilde{Y})=\rho$ et finalement r atteint son maximum $\rho>0$ sur C.

1.2 Enoncé et preuve du théorème de Perron-Frobenius

Théorème 7. *Perron-Frobenius*

Soit M une matrice de $\mathcal{M}_n(\mathbb{R})$ à coefficients positifs et primitive. Il existe X un vecteur propre pour M à coordonnées strictement positives et associé à $\rho\in\mathbb{R}^+$ tel que :

$\rightarrow dim(\mathrm{Ker}(M-\rho\, I_n))=1$

$\rightarrow \forall \lambda \in \mathrm{Sp}(M), |\lambda| \leqslant \rho$

On appelle rayon spectral de M le réel ρ mis en évidence.

Preuve.

Si M est primitive alors M est irréductible par le Lemme 4. M vérifie donc les hypothèses des Lemmes 5 et 6. Considérons alors ρ le maximum de la fonction r introduite au Lemme 6. On va montrer que $\rho \in \mathrm{Sp}(M)$ et que $|\lambda| \leqslant \rho$ si $\lambda \in \mathrm{Sp}(M)$.

Par définition, $\rho \geqslant 0$. De plus $r(1,0,...,0)=\sum_{1\leqslant k\leqslant n} m_{i,k} > 0$ donc $\rho > 0$. On pose maintenant $Y \in \tilde{C}$ tel que $r(Y)=\rho$. Supposons que le vecteur $MY-\rho\, Y$ soit non nul. D'après le Lemme 5, $Z=(I_n+M)^{n-1}Y$ est à coefficients strictement positifs. Ainsi pour ε assez petit, $A\,Z-(\rho+\varepsilon)\,Z$ est un vecteur à coefficients strictement positifs. On a donc $r(Z) \geqslant \rho+\varepsilon$, ce qui contredit la définition de ρ. Ainsi $MY=\rho\, Y$.

Soit maintenant $\lambda \in \mathrm{Sp}(M)$ et $X \in \mathbb{C}^n$ un vecteur propre associé. On a pour tout i, par inégalité triangulaire :

$|\lambda||X_i| \leqslant \sum_{1\leqslant j\leqslant n} m_{i,j}\,|X_j|$

Ainsi, en notant $\tilde{X}=(|X_1|,...,|X_n|)$, on a $|\lambda| \leqslant r(\tilde{X}) \leqslant \rho$. Et de ce fait $\lambda \in \mathrm{Sp}(M) \Rightarrow |\lambda| \leqslant \rho$

On va maintenant prouver que $\dim(\mathrm{Ker}(M-\rho\, I_n))=1$.

On suppose par l'absurde que $\dim(\mathrm{Ker}(M-\rho\, I_n)) \geqslant 2$ et on se donne deux vecteurs propres u et v linéairement indépendants à coefficients strictement positifs tels que $M\,u=\rho\,u$ et $M\,v=\rho\,v$. Ainsi, pour α et β dans $\mathbb{R}$, on a : $M(\alpha\,u+\beta\,v)=\rho\,(\alpha\,u+\beta\,v)$. Comme u et v sont linéairement indépendants, $\alpha\,u+\beta\,v \neq 0$ et c'est donc un vecteur propre pour ρ. Mais par le raisonnement précédent, tout vecteur propre pour ρ est forcément à coefficients strictement positifs, or on peut trouver un couple (α,β) tel que $\alpha\,u+\beta\,v$ ait une composante nulle au moins, c'est absurde. Ainsi $\dim(\mathrm{Ker}(M-\rho\, I_n))=1$.

1.3 Corollaires et application à la démographie

Corollaire 8. *Soit M une matrice de Leslie de taille n et ρ son rayon spectral donné par théorème de Perron-Frobenius. On note $X_0 \in \mathbb{R}_+^n$ le vecteur représentant la population initiale suivant n classes d'ages. Si on définit $(X_p)_{p\in\mathbb{N}}$ telle que $\forall p \in \mathbb{N}, X_{p+1}=M\,X_p$.*

Alors $\lim_{p\longrightarrow+\infty}\left(\frac{\sum_{1\leqslant k\leqslant n}(X_{p+1})_k}{\sum_{1\leqslant k\leqslant n}(X_p)_k}\right)=\rho$.

Corollaire 9. *Soit M une matrice de Leslie de taille n et $\rho > 0$ son rayon spectral donné par théorème de Perron-Frobenius. On note (X_n^i) l'effectif de la i-ème classe au temps n, tout cela étant défini par la relation de récurrence :*

$\forall i \in \{1,...,p\}, \forall n \in \mathbb{N},\ X_{n+1}^i = M\,X_n^i$

Alors l'un des trois cas suivants est envisageable:

$\boldsymbol{\rho > 1}$ *l'effectif de la population diverge exponentiellement vers $+\infty$.*

$\boldsymbol{\rho = 1}$ *l'effectif de la population converge vers une configuration stable égale au vecteur propre associé à ρ, noté X_0.*

$\boldsymbol{\rho < 1}$ *l'effectif de la population décroit exponentillement vers 0.*

Preuves des corollaires.

Le corollaire 9 se déduit directement du corollaire 8 en remarquant que la population à l'instant p s'écrit :

$P(p) = \sum_{1 \leqslant k \leqslant n} (X_p)_k \backsim \rho^p \sum_{1 \leqslant k \leqslant n} (X_0)_k = \exp(p \ln(\rho)) \sum_{1 \leqslant k \leqslant n} (X_0)_k$

Montrons alors le corollaire 8 et pour cela nous allons tout d'abord trigonaliser M, il existe donc $P \in \mathcal{M}_n(\mathbb{C})$ inversible telle que :

$M = P\,\mathrm{trig}(\rho := \lambda_1, \lambda_2, ..., \lambda_n)\,P^{-1}$ et on note T cette matrice triangulaire.

Alors, $X_{p+1} = PT^{p+1}\,P^{-1}\,X_0$ et $\rho\,X_p = PT^p\,P^{-1}\,X_0$.

En développant ces deux produit, on remarque que l'on peut retomber sur le membre de droite en factorisant celui de gauche par ρ et en servant du fait que ρ est valeur propre maximale stricte donc que tout les termes en $\left(\frac{\rho}{\lambda_k}\right)_{k \geqslant 2}$ sont strictement inférieurs à 1 donc que leur suite géométrique associée tend vers 0. Ainsi : $\frac{\sum_{1 \leqslant k \leqslant n} (X_{p+1})_k}{\rho \sum_{1 \leqslant k \leqslant n} (X_p)_k} \xrightarrow[p \to \infty]{} 1$.

D'où le résultat.

Interprétation.

Une fois déterminé le rayon spectral de la matrice de Leslie qui modélise l'évolution de population considérée, on applique le corollaire 9 et on peut ainsi déterminer le comportement asymptotique de cette population suivant les trois cas distingués.

2 Equation de McKendrick/Von Foerster

Dans cette partie on va établir une équation aux dérivées partielles (linéaire), reliant la densité de population aux taux de fertilité et de mortalité. Cela constitue donc une approche continue de la modélisation d'une évolution de population et nous verrons que cette équation se résout sous certaines conditions, ce qui nous permettra d'accéder à la valeur théorique de la population à tout instant et ce seulement en fonction des valeurs des taux.

2.1 Etablissement de l'équation

On pose $n(t, a)$ une densité de population dépendant du temps t et de l'âge a.

La population totale à l'instant t est donc $\int_0^\infty n(t, a)$. On note $\mu(t, a)$ et $\phi(t, a)$ les taux de mortalité et de fertilité.

Durant un temps dt, il meur $\mu(t, a)\,n(t, a)$ personnes.

On a donc : $dn(t, a) = \frac{\partial n}{\partial t}\,dt + \frac{\partial n}{\partial a}\,da = -\mu(t, a)\,n(t, a)\,dt$

On émet l'hypothèse que $da = dt$ *i.e* que le temps s'écoule de la même manière pour l'âge et pour le temps, on obtient l'équation de Mc-Kendrick/Von Foerster en simplifiant :

$$\frac{\partial n}{\partial t} + \frac{\partial n}{\partial a} = -\mu(t, a)\,n(t, a)$$

On émet à présent les hypothèses suivantes (paramètres constants + conditions aux bords):

1. Les taux de fertilité et de mortalité sont constants au cours du temps.

2. On se donne $n_0(a) = n(0,a)$ la distibution de population au temps initial

3. $\forall t \in \mathbb{R}_+ \quad n(t,0) = \int_0^\infty \phi(a) n(t,a) d\,a$ (naissances au temps t)

2.2 Résolution.

L'opérateur $X : n \to \frac{\partial n}{\partial t} + \frac{\partial n}{\partial a}$ est linéaire du premier ordre donc en posant le changement de variable :

$\begin{cases} \xi = a \\ \alpha = t - a \end{cases}$

On obtient: $\quad \frac{\partial n}{\partial \xi} = n(\xi, \alpha)\, \mu(\xi)$

En intégrant par rapport à ξ, cela donne l'existence d'une fonction f de la variable α telle que :

$n(\alpha, \xi) = f(\alpha) \exp\big(-\int_0^\xi \mu(a)\, d\,a\big)$

En repassant aux variables (t,a), cela donne : $\quad n(t,a) = f(t-a)\exp(-\int_0^a \mu(\varphi) d\varphi)$

(2.1.2) donne : $n_0(a) = f(-a) \exp(-\int_0^a \mu(\varphi)\, d\,\varphi)$

Et finalement : $\forall a \in \mathbb{R}_+ \quad f(-a) = n_0(a) \exp(\int_0^a \mu(\varphi) d\varphi)$

On pose à présent:

1. $M : a \longrightarrow \phi(a) \exp(-\int_0^a \mu(\varphi)\, d\varphi)$

2. Sous réserve d'existence, pour une fonction f : $L_f : \lambda > 0 \longrightarrow \int_0^{+\infty} f(t)\exp(-\lambda\, t) d\,t$

L_f est la transformée de Laplace de f, on admet l'existence d'une fonction notée L_f^{-1} définie sur $L_f(\mathbb{R}_+)$ telle que : $\forall \lambda \in \mathbb{R}_+ \quad L_f^{-1}(L_f(\lambda)) = f(\lambda)$.

On injecte la nouvelle forme de $n(t,a)$ dans (2.1.3), la fonction f doit donc vérifier pour $t \geq 0$: $f(t) = \int_0^\infty f(t-a)\, M(a) d\,a$

En remplacant f par son expression dans l'égalité précédente, on a pour $\lambda \in \mathbb{R}_+$:

$L_f(\lambda) = \int_0^\infty M(a) \exp(-\lambda\, a) \int_0^\infty f(t-a) \exp(-\lambda\,(t-a)) d\,t\, d\,a$

Par Chasles et changement de variable $\alpha = t - a$:

$L_f(\lambda) = \int_0^\infty M(a) \exp(-\lambda\, a) d\,a \big(\int_{-a}^0 f(\alpha) \exp(-\lambda\, \alpha) d\,\alpha + \int_0^\infty f(\alpha) \exp(-\lambda\, \alpha) d\alpha \big)$

$\qquad = L_f(\lambda)\, L_M(\lambda) + \int_0^\infty M(a) \exp(-\lambda\, a) d\,a \int_{-a}^0 f(\alpha) \exp(-\lambda\, \alpha) d\alpha$

Donc :

$L_f(\lambda)\,(1 - L_M(\lambda)) = \int_0^\infty M(a) \exp(-\lambda\, a) d\,a \int_{-a}^0 f(\alpha) \exp(-\lambda\, \alpha) d\alpha$

Dans notre cas on suppose que L_f est bien définie car si $M := \int_0^\infty M(a) d\,a < 1$ alors

$\forall \lambda \in \mathbb{R}_+ \quad 1 - L_M(\lambda) > 0$ et donc L_f s'écrit :

$$L_f(\lambda) = \frac{1}{1 - L_M(\lambda)} \int_0^\infty M(a) \exp(-\lambda\, a) d\,a \int_{-a}^0 f(\alpha) \exp(-\lambda\, \alpha) d\alpha$$

En utilisant L_f^{-1} la transformée de laplace inverse et la population à l'instant t s'écrit:

$$P(t) = \int_0^\infty n(t,a) d\,a = \int_0^\infty L_f^{-1}(L_f(t-a)) \exp\Big(-\int_0^a \mu(\varphi) d\varphi \Big) d\,a$$

2.3 Interprétation démographique

Il suffit donc de connaître f asymptotiquement pour avoir le comportement asymptotique de $n(t,a)$ et donc de la population. Il faut tout de même pour cela rappeller que n ne varie en fonction de a que sur un intervalle borné de la forme $I=[0,A]$ car il est raisonnable de supposer que n est nulle à partir d'une certain âge A.

Supposons que $L_f(\lambda)\,\lambda \xrightarrow[\lambda\to 0]{} l \in \mathbb{R}$, alors par théorème de la limite : $f(t) \xrightarrow[t\to+\infty]{} l$ et donc :

$$\forall 0 \leqslant a \leqslant A \quad n(t,a) \xrightarrow[t\to+\infty]{} l\exp(-\textstyle\int_0^a \mu(\varphi)d\varphi)$$

Ainsi : $\forall 0 \leqslant a \leqslant A, \forall \varepsilon > 0, \exists T > 0, \forall t \in \mathbb{R}: t > T \Rightarrow |n(t,a) - l\exp(-\int_0^a \mu(\varphi)d\varphi)| \leqslant \varepsilon$

Et donc : $\left|P(t) - l\int_0^A \exp(-\int_0^a \mu(\varphi)d\varphi)d\,a\right| \leqslant \int_0^A |n(t,a) - l\exp(-\int_0^a \mu(\varphi)d\varphi)| \leqslant \varepsilon\, A$

Ainsi, $P(t) \xrightarrow[t\to+\infty]{} l\int_0^A \exp(-\int_0^a \mu(\varphi)d\varphi)d\,a$

Par un raisonnement tout à fait similaire : $f(t) \xrightarrow[t\to+\infty]{} +\infty \Rightarrow P(t) \xrightarrow[t\to+\infty]{} +\infty$

Deuxième partie

Scripts Python

Chapitre 1

Interface graphique

GUI.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan  2 11:52:00 2018

@author: Utilisateur
"""
import tkinter as tk
from tkinter.filedialog import askopenfilename
#import matplotlib.pyplot as plt
#from mpl_toolkits.mplot3d.axes3d import Axes3D, get_test_data
#from matplotlib import cm
#import numpy as np
from time import time
#import psutil
#pid=os.getpid()
#ps=psutil.Process(pid)
from tkinter import ttk
from tkinter import messagebox
#import pygame
#from pygame.locals import *
#from time import sleep
#import os
#import random
#import gc
from modelediscret import *
from modelecontinu import *


class Fenetre(tk.Tk):
    def __init__(self,continu=3):
        self.continu=continu
        self.pop="populationfrance1998_2016.csv"
        self.mort="mortalitefrance1998_2015.csv"
        self.fec="feconditefrance1998_2015.csv"
        self.em="emigrationfrance1998_2015.csv"
        self.im="immigrationfrance1998_2015.csv"
        self.p=100
        self.annee=2007
        self.dt=1
        self.K=float("inf")
        tk.Tk.__init__(self)
        self.title("Leslie "+("continu" if continu else "discret")+
                   " | Dubard,Alcaraz, Monnier")
        self.partie1=tk.Frame(self,borderwidth=2,relief=tk.GROOVE)
        self.partie1.pack(side=tk.LEFT)
        self.partie2=tk.Frame(self,borderwidth=2,relief=tk.GROOVE)
        self.partie2.pack(side=tk.LEFT)
        #Partie1
        self.frame11=tk.Frame(self.partie1,borderwidth=2)
        self.frame11.pack(padx=10,pady=10)
        self.frame21=tk.Frame(self.partie1,borderwidth=2)
        self.frame21.pack(padx=10,pady=10)
        self.frame31=tk.Frame(self.partie1,borderwidth=2)
        self.frame31.pack(padx=10,pady=10)
        self.frame41=tk.Frame(self.partie1,borderwidth=2)
        self.frame41.pack(padx=10,pady=10)
        self.frame51=tk.Frame(self.partie1,borderwidth=2)
        self.frame51.pack(padx=10,pady=10)
        #partie2
        self.frame0=tk.Frame(self.partie2,borderwidth=2,relief=tk.GROOVE)
        self.frame0.pack(padx=10,pady=10)
```

```
self.frame1=tk.Frame(self.partie2,borderwidth=2,relief=tk.GROOVE)
self.frame1.pack(padx=10,pady=10)
self.frame2=tk.Frame(self.partie2,borderwidth=2,relief=tk.GROOVE)
self.frame2.pack(padx=10,pady=10)
self.frame3=tk.Frame(self.partie2,borderwidth=2,relief=tk.GROOVE)
self.frame3.pack(padx=10,pady=10)
#
tk.Label(self.frame11,text="fichier population initiale").pack()
tk.Label(self.frame21,text="fichier mortalité").pack()
tk.Label(self.frame31,text="fichier fécondité").pack()
tk.Label(self.frame41,text="fichier emigration").pack()
tk.Label(self.frame51,text="fichier immigration").pack()
self.fichier1=tk.Entry(self.frame11,width=50);self.fichier1.pack(
        side=tk.LEFT)
self.fichier1.insert(0,self.pop)
self.fichier2=tk.Entry(self.frame21,width=50);self.fichier2.pack(
        side=tk.LEFT)
self.fichier2.insert(0,self.mort)
self.fichier3=tk.Entry(self.frame31,width=50);self.fichier3.pack(
        side=tk.LEFT)
self.fichier3.insert(0,self.fec)
self.fichier4=tk.Entry(self.frame41,width=50);self.fichier4.pack(
        side=tk.LEFT)
self.fichier4.insert(0,self.em)
self.fichier5=tk.Entry(self.frame51,width=50);self.fichier5.pack(
        side=tk.LEFT)
self.fichier5.insert(0,self.im)
#
self.bouton1=tk.Button(self.frame11,text="parcourir",command=
                       self.parcourirpopulation)
self.bouton1.pack(side=tk.BOTTOM)
self.bouton2=tk.Button(self.frame21,text="parcourir",command=
                       self.parcourirmortalite)
self.bouton2.pack(side=tk.BOTTOM)
self.bouton3=tk.Button(self.frame31,text="parcourir",command=
                       self.parcourirfecondite)
self.bouton3.pack(side=tk.BOTTOM)
self.bouton4=tk.Button(self.frame41,text="parcourir",command=
                       self.parcouriremigration)
self.bouton4.pack(side=tk.BOTTOM)
self.bouton5=tk.Button(self.frame51,text="parcourir",command=
                       self.parcouririmmigration)
self.bouton5.pack(side=tk.BOTTOM)
#
self.v=tk.IntVar()
tk.Radiobutton(self.partie2,text="continu",variable=self.v,value=1,
               command=self.log).pack()
tk.Radiobutton(self.partie2,text="discret",variable=self.v,value=2,
               command=self.log).pack()
tk.Radiobutton(self.partie2,text="Leslie",variable=self.v,value=3,
               command=self.log).pack()
#
self.entree2=tk.Entry(self.frame1)
self.entree2.insert(0,str(self.p))
self.labelpasdetemps=tk.Label(self.frame0,text="Pas de temps dt : ")
self.entree0=tk.Entry(self.frame0)
self.entree0.insert(0,str(self.dt))
self.paslist=ttk.Combobox(self.frame0)
self.paslist["values"]=("ans","mois","jours","heure","minute",
            "seconde")
self.paslist.current(0)
```

```
        self.nbvaleurs_label=tk.Label(self.frame0,
                        text="nb de valeurs pour la résolution numérique : ")
        self.nbvaleurs_entry=tk.Entry(self.frame0)
        self.nbvaleurs_entry.insert(0,"1000")
        self.labelnbclasses=tk.Label(self.frame1,
                                     text="nb de classes d'âges : ")
        if continu==1:
            self.labelnbclasses.pack(side=tk.LEFT)
            self.entree2.pack(side=tk.LEFT)

        tk.Label(self.frame2,text="entrez l'année initiale :").pack(
                side=tk.LEFT)
        self.entree1=tk.Entry(self.frame2)
        self.entree1.pack(side=tk.LEFT)
        self.entree1.insert(0,str(self.annee))
        tk.Label(self.frame3,text="temps max :").pack(side=tk.LEFT)
        self.nentry=tk.Entry(self.frame3);self.nentry.pack(side=tk.LEFT)
        self.immigration_check_var=tk.BooleanVar()
        self.emmigration_check_var=tk.BooleanVar()

        tk.Checkbutton(self.frame3,var=self.immigration_check_var,
                       text="sans immigration").pack()
        tk.Checkbutton(self.frame3,var=self.emmigration_check_var,
                       text="sans emmigration").pack()
        self.rencontre_check_var=tk.BooleanVar()

        self.rencontrecheckbonton=tk.Checkbutton(self.frame3,
                                         var=self.rencontre_check_var,
                                         text="avec probabilité de rencontre")
        self.tauxfecmoyen=tk.BooleanVar()
        tk.Checkbutton(self.frame3,var=self.tauxfecmoyen,
                       text="avec taux de fécondité \"spéciaux\"").pack()
        self.transition_check=tk.BooleanVar()
        self.transitioncheckbouton=tk.Checkbutton(self.frame3,
                                         var=self.transition_check,
                                         text="modèle de transition",
                                         command=self.log)
        self.transitiontempsentry=tk.Entry(self.frame3)
        self.transitiontempslabel=tk.Label(self.frame3,
                                           text="durée de la transition :")
        if continu:
            self.rencontrecheckbonton.pack()
        self.logistique_check_var=tk.BooleanVar()
        tk.Checkbutton(self.frame3,var=self.logistique_check_var,
                       text="avec seuil logistique",command=self.log).pack()
        #
        self.logframe=tk.Frame(self.frame3)
        self.logframe.pack()
        self.loglabel=tk.Label(self.logframe,text="seuil de population : ")
        self.logentry=tk.Entry(self.logframe)
        self.logentry.insert(0,str(self.K))
        #
        self.Bouton=tk.Button(self.frame3,text="Tracer",command=self.callback)
        self.Bouton.pack()
        #
        self.mainloop()

    def log(self):
        if self.logistique_check_var.get():
            self.loglabel.pack(side=tk.LEFT)
            self.logentry.pack(side=tk.LEFT)
```

```
        else:
            self.loglabel.pack_forget()
            self.logentry.pack_forget()
        self.continu=self.v.get()

        if self.continu in [1,2]:
            if self.continu==1:
                self.nbvaleurs_label.pack(side=tk.LEFT)
                self.nbvaleurs_entry.pack(side=tk.LEFT)
                self.transitioncheckbouton.pack_forget()
                self.transitiontempslabel.pack_forget()
                self.transitiontempsentry.pack_forget()

            else:

                self.nbvaleurs_label.pack_forget()
                self.nbvaleurs_entry.pack_forget()
                self.transitioncheckbouton.pack(side=tk.LEFT)
            self.labelnbclasses.pack(side=tk.LEFT)
            self.entree2.pack(side=tk.LEFT)
            self.rencontrecheckbonton.pack()
        else:
            self.transitioncheckbouton.pack()
            self.transitiontempslabel.pack()
            self.transitiontempsentry.pack()
            self.nbvaleurs_label.pack_forget()
            self.nbvaleurs_entry.pack_forget()
            self.labelnbclasses.pack_forget()
            self.entree2.pack_forget()
            self.rencontrecheckbonton.pack_forget()
        if self.transition_check.get():
            self.transitiontempslabel.pack()
            self.transitiontempsentry.pack()
        else:
            self.transitiontempslabel.pack_forget()
            self.transitiontempsentry.pack_forget()

    def parcourir(self):
        filename =askopenfilename(parent = self,filetypes=(
                ("csv files","*.csv"),("All files","*.*")),
            title = 'Veuillez entrer un nom de fichier ')
        return filename
    def parcourirpopulation(self):#bouton parcourir fichier population
        self.pop=self.parcourir()
        self.fichier1.delete(0,tk.END)
        self.fichier1.insert(0,self.pop)
    def parcourirmortalite(self):#bouton parcourir fichier mortalité
        self.mort=self.parcourir()
        self.fichier2.delete(0,tk.END)
        self.fichier2.insert(0,self.mort)
    def parcourirfecondite(self):#bouton parcourir fichier fecondite
        self.fec=self.parcourir()
        self.fichier3.delete(0,tk.END)
        self.fichier3.insert(0,self.fec)
    def parcouriremigration(self):#bouton parcourir fichier emigration
        self.em=self.parcourir()
        self.fichier4.delete(0,tk.END)
        self.fichier4.insert(0,self.em)
    def parcouriritimmigration(self):#bouton parcourir fichier immigration
        self.im=self.parcourir()
        self.fichier5.delete(0,tk.END)
```

```
        self.fichier5.insert(0,self.im)

    def callback(self):

#         try:
        ti=time()
        self.annee=int(self.entree1.get())
        n=int(self.nentry.get())#pas de temps max
        self.p=int(self.entree2.get())
        self.K=float(self.logentry.get())
        if self.continu in [1,2]:
            self.dt=float(self.entree0.get())/({"ans":1,"mois":12,"jours":365,
                        "heure":365*24,"minute":365*24*60,
                        "seconde":365*24*60*60}[self.paslist.get()])

        #
        initiale,mortalite,fecondite,emmigration,immigration,
        fichierpopulationinitiale,femmes,mortalitehomme,mortalitefemme,hommes,
        immigrationhomme,immigrationfemme,emmigrationhomme,emmigrationfemme,
        fichierfemme,fichierhomme,ecartype=ouverturefichier(
                self.pop,self.mort,self.fec,self.em,self.im,self.annee,
                tauxfecmoyen=self.tauxfecmoyen.get())
        if self.immigration_check_var.get():
            print("!!! on ne prends pas en compte l'immigration")
            immigration=[0 for i in range(len(immigration))]
            immigrationhomme=[0 for i in range(len(immigration))]
            immigrationfemme=[0 for i in range(len(immigration))]
        if self.emmigration_check_var.get():
            print("!!! on ne prends pas en compte l'emmigration")
            emmigration=[0 for i in range(len(emmigration))]
            emmigrationhomme=[0 for i in range(len(emmigration))]
            emmigrationfemme=[0 for i in range(len(emmigration))]
        survie=matsurvie(mortalite,emmigration)
        if self.continu in [1,2]:
            resultat,yliste,yliste2,ecartliste,ecartlistetot,arret,hs=continu(
                    int(self.nbvaleurs_entry.get()),self.dt,self.p,hommes,
                    femmes,fecondite,mortalitehomme,mortalitefemme,
                    immigrationhomme,immigrationfemme,emmigrationhomme,
                    emmigrationfemme,n,self.annee,fichierpopulationinitiale,
                    fichierhomme,fichierfemme,
                    logistique=self.logistique_check_var.get(),
                    K=self.K,rencontre=self.rencontre_check_var.get(),
                    discret=(True if self.continu==2 else False),
                    tauxfecmoyen=self.tauxfecmoyen.get(),
                    ecartypefichier=ecartype,transition=(
                            False if self.continu!=2 else
                            self.transition_check.get()),
                            dureetransition=int(
                                    self.transitiontempsentry.get()))
            tracer(n,yliste,yliste2,ecartliste,ecartlistetot,arret,ti,hs,
                   tauxfecmoyen=self.tauxfecmoyen.get(),discret=(
                           True if self.v.get()==2 else False),
                           rencontre=self.rencontre_check_var.get())
        else:
            resultat,yliste,yliste2,ecartliste,ecartlistetot,arret,
            accroissement,courbefec,courbemort=matriciel(
                    matrice(initiale),Leslie(fecondite,survie),fecondite,
                    emmigration,mortalite,matimmigration(immigration),n,
                    self.annee,fichierpopulationinitiale,
                    logistique=self.logistique_check_var.get(),K=self.K,
                    transition=self.transition_check.get(),dureetransition=int(
```

```
                        self.transitiontempsentry.get() if
                        self.transitiontempsentry.get()!="" else "42"))
            enregistrement(tableau(yliste,yliste2,ecartliste,ecartlistetot,n,
                                   annee))
            print(yliste2)
            fig=plt.figure()
            fig.add_subplot(211)
            plt.plot([i for i in range(len(accroissement))],accroissement)
            plt.xlabel("temps(années)")
            plt.ylabel("accroissement naturel")
            plt.title("accroissement naturel")
            fig.add_subplot(212)
            plt.plot([i for i in range(len(courbefec))],courbefec,
                     label="fecondite")
            plt.plot([i for i in range(len(courbemort))],courbemort,
                     label="mortalite")
            plt.xlabel("temps(années)")

            plt.title("comparaison des taux moyen de fecondité et mortalité")
            plt.legend()
            plt.show()
            plt.grid()
            tracer(n,yliste,yliste2,ecartliste,ecartlistetot,arret,ti,
                   transition=self.transition_check.get(),
                   dureetransition=int(self.transitiontempsentry.get() if
                                self.transitiontempsentry.get()!="" else "42"))


if __name__=="__main__":
    F=Fenetre()
```

Chapitre 2

Modèle de Leslie

modelediscret.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 21 09:53:29 2017

@author: ldubard
r+te+tm+ts=1
r=1-dt/tmax

"""
#import tkinter as tk
#from tkinter.filedialog import askopenfilename
#from tkinter import messagebox
from time import time
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.axes3d import Axes3D, get_test_data
from matplotlib import cm
import numpy as np
from GUI import *
from ecart_type import *

###################################variables initiales#########################
monde=True
p=100#nb de classes d'âges
annee=2007
reste=[0 for x in range(p)]
pop="populationfrance1998_2016.csv"
mort="mortalitefrance1998_2015.csv"
fec="feconditefrance1998_2015.csv"
em="emigrationfrance1998_2015.csv"
im="immigrationfrance1998_2015.csv"

###################prise en charge des fichiers de statistiques###############

def ouverturefichier(pop,mort,fec,em,im,annee,tauxfecmoyen=False):
    with open(pop,"r") as fichier:
        liste=fichier.readlines()
        fichierpopulationinitiale=[(liste[x].split(";")) for x in range(len(liste))]
[10:10+100+3]
        colonne=fichierpopulationinitiale[0].index(str(annee))
        fichierfemme=[(liste[x].split(";")) for x in range(len(liste))][232:335]
        fichierhomme=[(liste[x].split(";")) for x in range(len(liste))][121:224]
        initiale=[0 for x in range(p)]
        femmes=[0 for x in range(p)]
        hommes=[0 for x in range(p)]
        for i in range(p-1):
            try:
                initiale[i]=float(fichierpopulationinitiale[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))
            except:
                initiale[i]=0
            try:
                femmes[i]=float(fichierfemme[i+2][colonne].replace(",",".").replace(
"\xa0","").replace(" ",""))
            except:
                femmes[i]=0
            try:
                hommes[i]=float(fichierhomme[i+2][colonne].replace(",",".").replace(
"\xa0","").replace(" ",""))
            except:
                hommes[i]=0
        for i in range(p-1,len(fichierpopulationinitiale)-2):
```

```
            try:
                initiale[p-1]+=float(fichierpopulationinitiale[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))
                femmes[p-1]+=float(fichierfemme[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))
                hommes[p-1]+=float(fichierhomme[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))
            except:
                continue
    with open(mort,"r") as fichier:
        liste=fichier.readlines()
        fichiermortalitehomme=[liste[x].split(";") for x in range(len(liste))]
[105:192]
        fichiermortalitefemme=[liste[x].split(";") for x in range(len(liste))]
[200:287]
        fichiermortalite=[liste[x].split(";") for x in range(len(liste))]
[10:97]#debut du tableau à la 11eme ligne du fichier csv et fin à la 97eme
        colonne=fichiermortalite[0].index(str(annee))
        mortalite=[0 for x in range(p)]
        mortalitehomme=[0 for x in range(p)]
        mortalitefemme=[0 for x in range(p)]
        for i in range(len(fichiermortalite)-1):
            mortalitefemme[i]=float(fichiermortalitefemme[i+1]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))
            mortalitehomme[i]=float(fichiermortalitehomme[i+1]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))
            mortalite[i]=float(fichiermortalite[i+1]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))
        for i in range(len(fichiermortalite)-1,p-1):
            mortalite[i]=float(fichiermortalite[len(fichiermortalite)-1]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))
            mortalitehomme[i]=float(fichiermortalitehomme[len(fichiermortalite)-1]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))
            mortalitefemme[i]=float(fichiermortalitefemme[len(fichiermortalite)-1]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))
        #


    with open(fec,"r") as fichier:
        ecartype=[0 for x in range(p)]
        liste=fichier.readlines()
        fichierfecondite=[liste[x].split(";") for x in range(len(liste))][10:48]
        colonne=fichierfecondite[0].index(str(annee) if not tauxfecmoyen else
"moyenne")
        fecondite=[0 for x in range(p)]
        for i in range(len(fichierfecondite)-2):
            if tauxfecmoyen:
                ecartype[i+15]=float(fichierfecondite[i+2][colonne
+1].replace(",",".").replace( "\xa0","").replace(" ",""))*femmes[i+15]/initiale[i
+15]
            fecondite[i+15]=float(fichierfecondite[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))*femmes[i+15]/
initiale[i+15]

        for i in range(len(fichierfecondite)-2,p-15):
            fecondite[i+15]=float(fichierfecondite[len(fichierfecondite)-1]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))*femmes[i+15]/
initiale[i+15]
            if tauxfecmoyen:
                ecartype[i+15]=float(fichierfecondite[len(fichierfecondite)-1]
[colonne+1].replace(",",".").replace( "\xa0","").replace(" ",""))*femmes[i+15]/
initiale[i+15]
```

```
    with open(em,"r") as fichier:
        liste=fichier.readlines()
        fichieremigration=[liste[x].split(";") for x in range(len(liste))][11:114]
        fichieremigrationhomme=[liste[x].split(";") for x in range(len(liste))]
[123:226]
        fichieremigrationfemme=[liste[x].split(";") for x in range(len(liste))]
[235:338]
        colonne=fichieremigration[0].index(str(annee))
        emmigration=[0 for x in range(p)]
        emmigrationhomme=[0 for x in range(p)]
        emmigrationfemme=[0 for x in range(p)]
        for i in range(p-1):
            emmigration[i]=float(fichieremigration[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ","").replace(":","0"))/
initiale[i] if initiale[i]!=0 else 0
            emmigrationhomme[i]=float(fichieremigrationhomme[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ","").replace(":","0"))/
hommes[i] if hommes[i]!=0 else 0
            emmigrationfemme[i]=float(fichieremigrationfemme[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ","").replace(":","0"))/
femmes[i] if femmes[i]!=0 else 0
        for i in range(p-1,len(fichieremigration)-2):
            emmigration[p-1]+=float(fichieremigration[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ","").replace(":","0"))/
initiale[p-1] if initiale[p-1]!=0 else 0
            emmigrationhomme[p-1]+=float(fichieremigrationhomme[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ","").replace(":","0"))/
hommes[p-1] if hommes[p-1]!=0 else 0
            emmigrationfemme[p-1]+=float(fichieremigrationfemme[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ","").replace(":","0"))/
femmes[p-1] if femmes[p-1]!=0 else 0
    with open(im,"r") as fichier:
        liste=fichier.readlines()
        fichierimmigration=[liste[x].split(";") for x in range(len(liste))][11:114]
        fichierimmigrationhomme=[liste[x].split(";") for x in range(len(liste))]
[123:226]
        fichierimmigrationfemme=[liste[x].split(";") for x in range(len(liste))]
[235:338]
        colonne=fichierimmigration[0].index(str(annee))
        immigration=[0 for x in range(p)]
        immigrationhomme=[0 for x in range(p)]
        immigrationfemme=[0 for x in range(p)]
        for i in range(p-1):
            div=initiale[i] if initiale[i]!=0 else 1
            divhomme=hommes[i] if hommes[i]!=0 else 1
            divfemme=femmes[i] if femmes[i]!=0 else 1
            try:
                immigration[i]=float(fichierimmigration[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))/div
            except:
                immigration[i]=0
            try:
                immigrationhomme[i]=float(fichierimmigrationhomme[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))/divhomme
            except:
                immigrationhomme[i]=0
            try:
                immigrationfemme[i]=float(fichierimmigrationfemme[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))/divfemme
            except:
```

```
                immigrationfemme[i]=0
        for i in range(p-1,len(fichierimmigration)-2):
            div=initiale[p-1] if initiale[p-1]!=0 else 1
            divhomme=hommes[p-1] if hommes[p-1]!=0 else 1
            divfemme=femmes[p-1] if femmes[p-1]!=0 else 1
            try:
                immigration[p-1]+=float(fichierimmigration[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))/div
            except:
                immigration[p-1]+=0

            try:
                immigrationhomme[p-1]+=float(fichierimmigrationhomme[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))/divhomme
            except:
                immigrationhomme[p-1]+=0
            try:
                immigrationfemme[p-1]+=float(fichierimmigrationfemme[i+2]
[colonne].replace(",",".").replace( "\xa0","").replace(" ",""))/divfemme
            except:
                immigrationfemme[p-1]+=0
    ########################################monde#########################################
    if monde:
        #fecondite=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,46.5,46.5,46.5,46.5,46.5,145.9,145
.9,145.9,145.9,145.9,143.8,143.8,143.8,143.8,143.8,96.7,96.7,96.7,96.7,96.7,49.9,49.
9,49.9,49.9,49.9,16.6,16.6,16.6,16.6,16.6,4,4,4,4,4,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
        fecondite=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.022739349,0.022739349,0.022739349,0
.022739349,0.022739349,0.0720137614,0.0720137614,0.0720137614,0.0720137614,0.0720137
614,0.0673922293,0.0673922293,0.0673922293,0.0673922293,0.0673922293,0.0456358619,0.
0456358619,0.0456358619,0.0456358619,0.0456358619,0.0244008262,0.0244008262,0.024400
8262,0.0244008262,0.0244008262,0.0080176719,0.0080176719,0.0080176719,0.0080176719,0
.0080176719,0.0018957219,0.0018957219,0.0018957219,0.0018957219,0.0018957219]+[1/100
0]*(100-49)
        #print(len(fecondite))
        initiale=[673650,673650,673650,673650,673650,644476,644476,644476,644476,644
476,612407,612407,612407,612407,612407,593639,593639,593639,593639,593639,600867,600
867,600867,600867,600867,612403,612403,612403,612403,612403,553200,553200,553200,553
200,553200,500817,500817,500817,500817,500817,486172,486172,486172,486172,486172,455
906,455906,455906,455906,455906,403243,403243,403243,403243,403243,340228,340228,340
228,340228,340228,294105,294105,294105,294105,294105,216506,216506,216506,216506,216
506,153908,153908,153908,153908,153908,115532,115532,115532,115532,115532,71831,7183
1,71831,71831,71831,37295,37295,37295,37295,37295,13475,13475,13475,13475,13475,2917
*4/5,2917*4/5,2917*4/5,2917*4/5,2917/5+434]
        mortalite=[34081,34081,34081,34081,34081,3245,3245,3245,3245,3245,2839,2839,
2839,2839,2839,3524,3524,3524,3524,3524,4728,4728,4728,4728,4728,5270,5270,5270,5270
,5270,5667,5667,5667,5667,5667,6372,6372,6372,6372,6372,7451,7451,7451,7451,7451,921
8,9218,9218,9218,9218,11716,11716,11716,11716,11716,15204,15204,15204,15204,15204,19
099,19099,19099,19099,19099,22449,22449,22449,22449,22449,27172,27172,27172,27172,27
172,30606,30606,30606,30606,30606,29130,29130,29130,29130,29130,22275,22275,22275,22
275,22275,11759,11759,11759,11759,11759,0.9*2917*4/5,0.9*2917*4/5,0.9*2917*4/5,0.9*2
917*4/5,2917/5+434]
        #fecondite=[fecondite[i]*2/1000 for i in range(len(fecondite))]
        initiale=[initiale[i]*1000/(5 if i<95 else (4 if i<99 else 1)) for i in
range(len(initiale))]
        mortalite=[mortalite[i]/5*1000/(5 if i<95 else (4 if i<99 else 1))/
initiale[i] for i in range(len(mortalite))]
        #mortalite[95:]=[0.8]*4+[1]
        immigration=[0 for i in range(len(immigration))]
        emmigration=[0 for i in range(len(emmigration))]
    #print(mortalite)
```

```
    return
initiale,mortalite,fecondite,emmigration,immigration,fichierpopulationinitiale,femme
s,mortalitehomme,mortalitefemme,hommes,immigrationhomme,immigrationfemme,emmigration
homme,emmigrationfemme,fichierfemme,fichierhomme,ecartype


################################################################################



######################LES MATRICES X(0),I et L#############################
def matsurvie(mortalite,emmigration,reste=[0 for x in range(p)]):#population qui
passe à la classe suivante
    """valeur par defaut de reste est 0 car pas de temps = durée d'une classe
d'âge"""
    survie=[(1-mortalite[x]-emmigration[x]-reste[x]) for x in range(p-1)]
    return survie

def matimmigration(immigration):#matrice I
    matrice=np.zeros((p,p))
    for i in range(p):#complexité : O(p)
        matrice[i,i]=immigration[i]
    return matrice

def matrice(liste):#matrice X(0)
    matrice=np.array([[x] for x in liste])
    return (matrice)

def Leslie(fecondite,survie,reste=[0 for x in range(p)]):#matrice L
    matrice=[[0 for x in range(p)] for y in range(p-1)]
    for i in range(p-1):#complexité : O(p)
        matrice[i][i]=survie[i]
    matrice=np.concatenate((np.array([fecondite]),np.array(matrice)),axis=0).tolist(
)
    for i in range(p):#complexité : O(p)
        matrice[i][i]+=reste[i]
    matrice=np.array(matrice)
    return (matrice)

def fonction(matrice,Leslie,immigration,logistique,K):
    if logistique:
        for j in range(p):
                Leslie[0,j]*=(1-sum([matrice[0][j] for j in
range(len(matrice[0]))])/K) if (K-sum([matrice[0][j] for j in
range(len(matrice[0]))]))>0 else 0
    return Leslie+immigration
######################écarts relatifs#####################################
def ecart(resultat,yliste2,pastemps,annee,populationinitiale):
    arret=pastemps+1#pour la population totale
    a=True
    colonne=populationinitiale[0].index(str(annee))
    ecarttot=[0 for i in range(pastemps+1)]
    ecartrelatif=[[0 for i in range(pastemps+1)] for j in range(p)]
    for i in range(pastemps+1):#complexité : O(p)
        try:
            tot=populationinitiale[1][colonne+i].replace(",",".").replace(
"\xa0","").replace(" ","")
            ecarttot[i]=abs(float(tot)-yliste2[i])/float(tot)
        except:
            ecarttot[i]="pas d'info"
```

```
            if a:
                arret=i
                a=False
        for j in range(p-1):#complexité : O(p)
            try:
                init=populationinitiale[j+2][colonne+i].replace(",",".").replace(
"\xa0","").replace(" ","")
                ecartrelatif[j][i]=abs(float(init)-resultat[j][i])/float(init)
            except:
                ecartrelatif[j][i]="pas d'info"
        init=0
        for j in range(p-1,len(populationinitiale)-2):
            try:
                init+=float(populationinitiale[j+2][colonne
+i].replace(",",".").replace( "\xa0","").replace(" ",""))
            except:
                pass
        if float(init)!=0:
            ecartrelatif[p-1][i]=abs(float(init)-resultat[p-1][i])/float(init)
        else:
            ecartrelatif[p-1][i]="pas d'info"
    return ecartrelatif,ecarttot,arret

##############################################################################
def
tracer(n,yliste,yliste2,ecartliste,ecartlistetot,arret,ti,hs=None,tauxfecmoyen=None,
discret=True,rencontre=False,transition=False,dureetransition=42):#complexité :
O(np)
    #print(rencontre)
    fig = plt.figure()
    #print(hs)
    fig.suptitle("Evolution "+("de la population mondiale" if monde else "d'une
population")+" en "+str(p)+" classes d'âges selon le modèle "+("de Leslie" if
hs==None else "de notre composition")+"\n ("+("temps discret " if discret else
"temps continu : "+str(hs)+" valeurs ")+(("avec proba de rencontre") if
rencontre==True else ("") )+((" avec taux \"spéciaux\"" ) if tauxfecmoyen==True else
(""))+((" avec une transition de "+str(dureetransition)+" ans" ) if transition==True
else (""))+")")
    ax = fig.add_subplot(221, projection='3d')
    if hs==None or discret:
        #print(n)
        xliste=[i for i in range(n+1)]
    else:
        xliste=np.linspace(0,n,hs)
    x=[i for i in range(len(yliste[0]))]
    y=[i for i in range(len(yliste))]
    z=np.array([[yliste[i][j] for j in range(len(yliste[i]))] for i in
range(len(yliste))])
    X,Y=np.meshgrid(x,y)
    ax.plot_wireframe(X,Y,z,rstride=5,cstride=5)
    ax.set_xlabel("pas de temps n")
    ax.set_ylabel("classe d'âge")
    ax.set_zlabel("individus")
    ax.legend()
    ax = fig.add_subplot(222)
    ax.plot(xliste,yliste2,label="population totale")
    ax.set_xlabel("pas de temps n")
    ax.set_ylabel("nb d'individu")
    ax.legend()
    ax.grid()
    ##ecarts relatifs
```

```
    #pour chaque classe
    ax = fig.add_subplot(223,projection="3d")
    x=[i for i in range(arret)]
    y=[i for i in range(len(ecartliste) if len(ecartliste)==100 else
len(ecartliste)-30)]
    z=np.array([[ecartliste[i][j] for j in range(arret)] for i in range(len(y))])
    X,Y=np.meshgrid(x,y)
    surf=ax.plot_surface(X, Y, z, rstride=2, cstride=3,
cmap=cm.coolwarm,linewidth=50, antialiased=True)
    ax.set_zlim(-0.01, max([max([ecartliste[i][j] for j in range(arret)])for i in
range(len(y))]))
    fig.colorbar(surf, shrink=0.5, aspect=10)
    ax.set_xlabel("annee")
    ax.set_ylabel("classe d'âge")
    ax.set_zlabel("erreur relative")
    ax.legend()
    ax.grid()
    ##pour la population totale
    ax = fig.add_subplot(224)
    xliste=[i for i in range(len(ecartlistetot))]
    ax.plot(xliste[:arret],ecartlistetot[:arret],label="erreur relative")
    ax.set_xlabel("annee")
    ax.set_ylabel("erreur relative")
    ax.legend()
    ax.grid()
    print("erreur relative totale : ", ecartlistetot)
    tf=time()
    print("temps d'execution : ",tf-ti,"s")
    plt.show()
##########################################################################################
##################
def
matriciel(matrice,Leslie,fecondite,emmigration,mortalite,immigration,n,annee,fichier
populationinitiale,logistique=False,K=float("inf"),transition=False,dureetransition=
42):#complexité : O(np^3)
    #global fichierpopulationinitiale
    pondere=False
    retard=int(dureetransition*0.1)
    coeffdroitetransitionfecondite=-(max(fecondite)-(15/1000))/dureetransition

    l=1/(85-5)*np.log(mortalite[84]/mortalite[4])
    #coeffdroitetransitionmortalite=-(mortalite[5]/np.exp(5*l)-0.0003*99*l/
((np.exp(99*l)-1)))/dureetransition
    #print(mortalite[5])
    coeffdroitetransitionmortalite=-((mortalite[5]-0.0001904520061)/np.exp(5*l))/
dureetransition
    if pondere:
        accroissement=np.array([sum([(Leslie[0][j]-mortalite[j])*matrice[j,0] for j
in range(len(Leslie))])/sum(matrice[:,0]) for i
                                in range(n)])
        courbefec=np.array([sum([(Leslie[0][j])*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0]) for i
                                in range(n)])
        courbemort=np.array([sum([(mortalite[j])*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0]) for i
                                in range(n)])
    else:
        accroissement=np.array([sum([(Leslie[0][j]-mortalite[j]) for j in
range(len(Leslie))])/len(Leslie) for i
                                in range(n)])
        courbefec=np.array([sum([(Leslie[0][j]) for j in range(len(Leslie))])/
```

```
len(Leslie) for i
                                    in range(n)])
        courbemort=np.array([sum([(mortalite[j]) for j in range(len(Leslie))])/
len(Leslie) for i
                                    in range(n)])
    yliste=np.array([[0 for i in range(n+1)] for x in
range(matrice.shape[0])],dtype="float64")
    yliste2=np.array([0 for i in range(n+1)],dtype="float64")
    for x in range(len(yliste)):#complexité : O(p)
        yliste[x][0]=matrice[x,0] #initialisation |
        yliste2[0]+=matrice[x,0]
    if logistique and K!=float("inf"):
        for i in range (n):#compléxité : O(p^3n)
            if transition:
                if i>=retard and i<=dureetransition+retard:
                    Leslie[0]=[Leslie[0][j]+coeffdroitetransitionfecondite*np.exp(-
(j-esperance)**2/(2*variance)) for j in range(len(Leslie[0]))]
                if i<=dureetransition:
                    for j in range(len(Leslie)-1):

                        Leslie[j+1][j]=(1-(mortalite[i]
+coeffdroitetransitionmortalite*i*np.exp(l*(i)))-emmigration[i])
                        if pondere:
                            accroissement[i]=sum([(Leslie[0][j]-mortalite[j]-
coeffdroitetransitionmortalite*i*np.exp(l*j))*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                            courbefec[i]=sum([(Leslie[0][j])*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                            courbemort[i]=sum([(mortalite[j]
+coeffdroitetransitionmortalite*i*np.exp(l*j))*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                        else:
                            accroissement[i]=sum([(Leslie[0][j]-mortalite[j]-
coeffdroitetransitionmortalite*i*np.exp(l*j)) for j in range(len(Leslie))])/
len(Leslie)
                            courbefec[i]=sum([(Leslie[0][j]) for j in
range(len(Leslie))])/len(Leslie)
                            courbemort[i]=sum([(mortalite[j]
+coeffdroitetransitionmortalite*i*np.exp(l*j)) for j in range(len(Leslie))])/
len(Leslie)
                else:
                    if pondere:
                        accroissement[i]=sum([(Leslie[0][j]-mortalite[j]-
coeffdroitetransitionmortalite*dureetransition*np.exp(l*j))*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                        courbefec[i]=sum([(Leslie[0][j])*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                        courbemort[i]=sum([(mortalite[j]
+coeffdroitetransitionmortalite*dureetransition*np.exp(l*j))*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                    else:
                        accroissement[i]=sum([(Leslie[0][j]-mortalite[j]-
coeffdroitetransitionmortalite*dureetransition*np.exp(l*j))for j in
range(len(Leslie))])/len(Leslie)
                        courbefec[i]=sum([(Leslie[0][j]) for j in
range(len(Leslie))])/len(Leslie)
                        courbemort[i]=sum([(mortalite[j]
+coeffdroitetransitionmortalite*dureetransition*np.exp(l*j)) for j in
range(len(Leslie))])/len(Leslie)
            else:
                if pondere:
```

```
                    accroissement[i]=sum([(Leslie[0][j]-mortalite[j])*matrice[j,0]
for j in range(len(Leslie))])/sum(matrice[:,0])
                    courbefec[i]=sum([(Leslie[0][j])*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                    courbemort[i]=sum([(mortalite[j])*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                else:
                    accroissement[i]=sum([(Leslie[0][j]-mortalite[j]) for j in
range(len(Leslie))])/len(Leslie)
                    courbefec[i]=sum([(Leslie[0][j]) for j in range(len(Leslie))])/
len(Leslie)
                    courbemort[i]=sum([(mortalite[j]) for j in range(len(Leslie))])/
len(Leslie)
            matrice=np.dot(fonction(matrice,Leslie,immigration,logistique,K),matrice
)#complexité : O(p^3) au pire et O(p^2) au mieux
            for x in range(len(yliste)):# complexité : O(p)
                yliste[x][i+1]=matrice[x,0]
                yliste2[i+1]+=matrice[x,0]
    else:
        for i in range (n):#compléxité : O(p^3n)
            if transition:
                if i>=retard and i<=dureetransition+retard:
                    Leslie[0]=[Leslie[0][j]+coeffdroitetransitionfecondite*np.exp(-
(j-esperance)**2/(2*variance)) for j in range(len(Leslie[0]))]
                if i<=dureetransition:
                    for j in range(len(Leslie)-1):

                        Leslie[j+1][j]=(1-(mortalite[i]
+coeffdroitetransitionmortalite*i*np.exp(l*(i)))-emmigration[i])
                        if pondere:
                            accroissement[i]=sum([(Leslie[0][j]-mortalite[j]-
coeffdroitetransitionmortalite*i*np.exp(l*j))*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                            courbefec[i]=sum([(Leslie[0][j])*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                            courbemort[i]=sum([(mortalite[j]
+coeffdroitetransitionmortalite*i*np.exp(l*j))*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                        else:
                            accroissement[i]=sum([(Leslie[0][j]-mortalite[j]-
coeffdroitetransitionmortalite*i*np.exp(l*j)) for j in range(len(Leslie))])/
len(Leslie)
                            courbefec[i]=sum([(Leslie[0][j]) for j in
range(len(Leslie))])/len(Leslie)
                            courbemort[i]=sum([(mortalite[j]
+coeffdroitetransitionmortalite*i*np.exp(l*j)) for j in range(len(Leslie))])/
len(Leslie)
                else:
                    if pondere:
                        accroissement[i]=sum([(Leslie[0][j]-mortalite[j]-
coeffdroitetransitionmortalite*dureetransition*np.exp(l*j))*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                        courbefec[i]=sum([(Leslie[0][j])*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                        courbemort[i]=sum([(mortalite[j]
+coeffdroitetransitionmortalite*dureetransition*np.exp(l*j))*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                    else:
                        accroissement[i]=sum([(Leslie[0][j]-mortalite[j]-
coeffdroitetransitionmortalite*dureetransition*np.exp(l*j))for j in
range(len(Leslie))])/len(Leslie)
```

```
                        courbefec[i]=sum([(Leslie[0][j]) for j in
range(len(Leslie))])/len(Leslie)
                        courbemort[i]=sum([(mortalite[j]
+coeffdroitetransitionmortalite*dureetransition*np.exp(l*j)) for j in
range(len(Leslie))])/len(Leslie)
            else:
                if pondere:
                    accroissement[i]=sum([(Leslie[0][j]-mortalite[j])*matrice[j,0]
for j in range(len(Leslie))])/sum(matrice[:,0])
                    courbefec[i]=sum([(Leslie[0][j])*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                    courbemort[i]=sum([(mortalite[j])*matrice[j,0] for j in
range(len(Leslie))])/sum(matrice[:,0])
                else:
                    accroissement[i]=sum([(Leslie[0][j]-mortalite[j]) for j in
range(len(Leslie))])/len(Leslie)
                    courbefec[i]=sum([(Leslie[0][j]) for j in range(len(Leslie))])/
len(Leslie)
                    courbemort[i]=sum([(mortalite[j]) for j in range(len(Leslie))])/
len(Leslie)
            matrice=np.dot(fonction(matrice,Leslie,immigration,logistique,K),matrice
)#complexité : O(p^3) au pire et O(p^2) au mieux
            for x in range(len(yliste)):# complexité : O(p)
                yliste[x][i+1]=matrice[x,0]
                yliste2[i+1]+=matrice[x,0]
    ecartliste,ecartlistetot,arret=ecart(yliste,yliste2,n,annee,fichierpopulationini
tiale)
    return
matrice,yliste,yliste2,ecartliste,ecartlistetot,arret,accroissement,courbefec,courbe
mort
################################################################################

####################enregistrement des résultats en csv#####################
def tableau(yliste,yliste2,ecartliste,ecarttot,n,annee):#complexité : O(np)
    tableau=[[0 for x in range(2*n+3)] for i in range(len(yliste)+2)]
    tableau[0][0]="AGE/TIME"
    tableau[1][0]="TOTAL"
    for j in range(1,len(tableau[0])):#complexité : O(n)
        if (j-1)%2==0:
            tableau[0][j]=str(annee+(j-1)//2)
            tableau[1][j]=str(yliste2[(j-1)//2])
        else:
            tableau[0][j]="ecart relatif"
            tableau[1][j]=str(ecarttot[j//2-1])
    for i in range(2,len(tableau)):#complexité : O(pn)
        tableau[i][0]=str(i-2)
        for j in range(1,len(tableau[i])): #complexité : O(n)
            if (j-1)%2==0:
                tableau[i][j]=str(yliste[i-2][(j-1)//2])
            else:
                tableau[i][j]=str(ecartliste[i-2][j//2-1])
    return tableau

def enregistrement(tableau):#complexité : O(n)
    resultat=[";".join(tableau[x])+"\n" for x in range(len(tableau))]
    fichier=open("resultatsimulation.csv","w")
    fichier.writelines(resultat)
    fichier.close()
```

Chapitre 3

Modèles discret et continu

modelecontinu.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan  2 12:33:06 2018

@author: Utilisateur
n=100/p
r+s+m+e=1
r=(1-dt/n)*(1-m-e)
s=dt/n*(1-m-e)

X'(t)=f(X,t)=(X(t+dt)-X(t))/dt
"""
#import tkinter as tk
#from tkinter.filedialog import askopenfilename
#import matplotlib.pyplot as plt
#from mpl_toolkits.mplot3d.axes3d import Axes3D, get_test_data
#from matplotlib import cm
import numpy as np
#from time import time
#import psutil
#pid=os.getpid()
#ps=psutil.Process(pid)
#from tkinter import messagebox
#import pygame
#from pygame.locals import *
#from time import sleep
#import os
import random as rd
#import gc
from ecart_type import *
#from modelediscret import *
nb=10000
def Euler(f,y0,t0,tf,n):
    h=abs(tf-t0)/(n-1)
    #n=1+abs(tf-t0)//dt
    t=np.linspace(t0,tf,n)
    Y=np.zeros((n,len(y0)))
    Y[0]=y0
    for i in range(n-1):
        Y[i+1]=Y[i]+h*f(Y[i],t[i])
    return t,Y

def RK4(f,y0,t0,tf,n):
    h=abs(tf-t0)/(n-1)
    t=np.linspace(t0,tf,n)
    Y=np.zeros((n,len(y0)))
    Y[0]=y0
    for i in range(n-1):
        k1=f(Y[i],t[i])
        k2=f(Y[i]+h/2*k1,t[i]+h/2)
        k3=f(Y[i]+h/2*k2,t[i]+h/2)
        k4=f(Y[i]+h*k3,t[i]+h)
        Y[i+1]=Y[i]+h/6*(k1+2*k2+2*k3+k4)
    return t,Y

def discretfct(f,y0,n):
    Y=np.zeros((n+1,len(y0)))
    Y[0]=y0
    t=0
    for i in range(n):
        t+=1
```

```
        Y[i+1]=f(Y[i],t)
    return Y

def
ecartcontinu(dt,resultat,yliste2,fichierhomme,fichierfemme,populationinitiale,n,p,an
nee,hs=nb,discret=False):
    arret=int(n*dt)+1
    a=True
    hs=len(yliste2)
    #t=np.linspace(0,n,hs)
    if not discret:
        yliste2=[yliste2[int(i*(hs-1)/(dt*n))] for i in range(int(n*dt)+1)]
        resultat=[[resultat[int(i*(hs-1)/(dt*n))][j] for i in range(int(n*dt)+1)]
for j in range(p*2)]
    colonne=populationinitiale[0].index(str(annee))
    ecarttot=[0 for i in range(int(n*dt)+1)]
    ecartrelatif=[[0 for i in range(int(n*dt)+1)] for j in range(p*2)]
    #initiale=[[0 for j in range(p)] for i in range(p)]
    for i in range(int(n*dt)+1):#complexité : O(p)
        try:
            tot=populationinitiale[1][colonne+i].replace(",",".").replace(
"\xa0","").replace(" ","")
            ecarttot[i]=abs(float(tot)-yliste2[i])/float(tot)
        except:
            ecarttot[i]="pas d'info"
            if a:
                arret=i
                a=False
        #####
        nbfemme=[0 for x in range(100)]
        nbhomme=[0 for x in range(100)]
        for j in range(100-1):
            try:
                nbfemme[j]=float(fichierfemme[j+2][colonne
+i].replace(",",".").replace( "\xa0","").replace(" ",""))
            except:
                nbfemme[j]=0
            try:
                nbhomme[j]=float(fichierhomme[j+2][colonne
+i].replace(",",".").replace( "\xa0","").replace(" ",""))
            except:
                nbhomme[j]=0
        for j in range(100-1,len(populationinitiale)-2):
            try:
                nbfemme[100-1]+=float(fichierfemme[j+2][colonne
+i].replace(",",".").replace( "\xa0","").replace(" ",""))
                nbhomme[100-1]+=float(fichierhomme[j+2][colonne
+i].replace(",",".").replace( "\xa0","").replace(" ",""))
            except:
                pass

        y0=[0 for j in range(p*2)]
        for k in range(p):
            x1,x2=k*100//p,(k+1)*100//p
            if k==p-1:
                x2=100

            y0[2*k]=sum(nbhomme[x1:x2])
            y0[2*k+1]=sum(nbfemme[x1:x2])
```

```
        for j in range(2*p):#complexité : O(p)
            if float(y0[j])!=0:
                ecartrelatif[j][i]=abs(float(y0[j])-resultat[j][i])/float(y0[j]) if
not discret else abs(float(y0[j])-resultat[i][j])/float(y0[j])
            else:
                ecartrelatif[j][i]="pas d'info"


    return ecartrelatif,ecarttot,arret

##############################################################################
def
X_tplusdt(t,X,dt,fecondite,immigration,reste,survie,rencontre,logistique,K,tauxfecmo
yen,ecartype,transition,coeffdroitetransitionfecondite,coeffdroitetransitionmortalit
eh,coeffdroitetransitionmortalitef,p,emmigration,mortalite,lh,lf,dureetransition):
    matrice=[0 for i in range(len(X))]
    for i in range(len(matrice)):
        matrice[i]=(reste[i]+immigration[i])*X[i]+(survie[i-2]*X[i-2] if i>1 else 0)
    if transition and t<=dureetransition:
        fecondite=[fecondite[j]-coeffdroitetransitionfecondite*t*np.exp(-(j-
esperance)**2/(2*variance)) for j in range(len(fecondite))]
        survie=[dt*(p/100)*(1-(mortalite[i]-(coeffdroitetransitionmortaliteh if i
%2==0 else coeffdroitetransitionmortalitef)*t*np.exp((lh if i%2==0 else lf)*(i//
2)))-emmigration[i]) for i in range(len(survie))]
    if rencontre:
        if logistique and K!=float("inf"):
            for j in range(len(X)//2):
                matrice[0]+=((fecondite[j]+(rd.random()*2-1)*ecartype[j])/
2)*(X[2*j]*X[2*j+1]/(X[2*j]+X[2*j+1]))*((1-sum([X[i] for i in range(len(X))])/K) if
K-sum([X[i] for i in range(len(X))])>0 else 0)
                matrice[1]+=((fecondite[j]+(rd.random()*2-1)*ecartype[j])/
2)*(X[2*j]*X[2*j+1]/(X[2*j]+X[2*j+1]))*((1-sum([X[i] for i in range(len(X))])/K) if
K-sum([X[i] for i in range(len(X))])>0 else 0)
        else:
            for j in range(len(X)//2):
                matrice[0]+=((fecondite[j]+(rd.random()*2-1)*ecartype[j])/
2)*(X[2*j]*X[2*j+1]/(X[2*j]+X[2*j+1]))
                matrice[1]+=((fecondite[j]+(rd.random()*2-1)*ecartype[j])/
2)*(X[2*j]*X[2*j+1]/(X[2*j]+X[2*j+1]))
    else:
        if logistique and K!=float("inf"):
            for j in range(len(X)//2):
                matrice[0]+=((fecondite[j]+(rd.random()*2-1)*ecartype[j])/2)*X[2*j
+1]*((1-sum([X[i] for i in range(len(X))])/K) if K-sum([X[i] for i in
range(len(X))])>0 else 0)
                matrice[1]+=((fecondite[j]+(rd.random()*2-1)*ecartype[j])/2)*X[2*j
+1]*((1-sum([X[i] for i in range(len(X))])/K) if K-sum([X[i] for i in
range(len(X))])>0 else 0)
        else:
            for j in range(len(X)//2):
                matrice[0]+=((fecondite[j]+(rd.random()*2-1)*ecartype[j])/2)*X[2*j
+1]
                matrice[1]+=((fecondite[j]+(rd.random()*2-1)*ecartype[j])/2)*X[2*j
+1]
    return matrice


def
continu(nb,dt,p,nbhomme,nbfemme,fec,mortalitehomme,mortalitefemme,immigrationhomme,i
mmigrationfemme,emmigrationhomme,emmigrationfemme,n,annee,fichierpopulationinitiale,
fichierhomme,fichierfemme,logistique=False,K=float("inf"),rencontre=True,discret=Fal
```

```
se,tauxfecmoyen=False,ecartypefichier=None,transition=False,dureetransition=42):
    assert K>sum(nbhomme)+sum(nbfemme),"!!! seuil dépassé initialement !!!"
    y0=[0 for i in range(p*2)]
    fecondite=[0 for i in range(p)]
    immigration=[0 for i in range(p*2)]
    reste=[0 for i in range(p*2)]
    survie=[0 for i in range(p*2)]
    mortalite=[0 for i in range(p*2)]
    emmigration=[0 for i in range(p*2)]
    ecartype=[0 for i in range(p)]
    coeffdroitetransitionfecondite=8.2029E-4*42/dureetransition


    for k in range(p):
        x1,x2=k*100//p,(k+1)*100//p
        if k==p-1:
            x2=100
        y0[2*k]=sum(nbhomme[x1:x2])
        y0[2*k+1]=sum(nbfemme[x1:x2])

        fecondite[k]=dt*sum([fec[i]*(nbhomme[i]+nbfemme[i])/nbfemme[i] for i in
range(x1,x2)])/(x2-x1)*(((y0[2*k]+y0[2*k+1])/y0[2*k]) if rencontre else 1)*((1/(1-
(sum(nbhomme)+sum(nbfemme))/K)) if logistique and K!=float("inf") else 1)
        if ecartypefichier!=None:
            ecartype[k]=dt*sum([ecartypefichier[i]*(nbhomme[i]+nbfemme[i])/
nbfemme[i] for i in range(x1,x2)])/(x2-x1)*(((y0[2*k]+y0[2*k+1])/y0[2*k]) if
rencontre else 1)*((1/(1-(sum(nbhomme)+sum(nbfemme))/K)) if logistique and K!
=float("inf") else 1)
        immigration[2*k]=dt*sum([immigrationhomme[i] for i in range(x1,x2)])/(x2-x1)
        immigration[2*k+1]=dt*sum([immigrationfemme[i] for i in range(x1,x2)])/(x2-
x1)
        mortalite[2*k]=sum([mortalitehomme[i] for i in range(x1,x2)])/(x2-x1)
        mortalite[2*k+1]=sum([mortalitefemme[i] for i in range(x1,x2)])/(x2-x1)
        emmigration[2*k]=sum([emmigrationhomme[i] for i in range(x1,x2)])/(x2-x1)
        emmigration[2*k+1]=sum([emmigrationfemme[i] for i in range(x1,x2)])/(x2-x1)
        reste[2*k]=dt*(1-p/100)*(1-mortalite[2*k]-emmigration[2*k])
        survie[2*k]=dt*(p/100)*(1-mortalite[2*k]-emmigration[2*k])
        reste[2*k+1]=dt*(1-p/100)*(1-mortalite[2*k+1]-emmigration[2*k+1])
        survie[2*k+1]=dt*(p/100)*(1-mortalite[2*k+1]-emmigration[2*k+1])
    lh=1/(85-5)*np.log(mortalitehomme[84]/mortalitehomme[4])
    coeffdroitetransitionmortaliteh=5.2349E-4*99*lh/(np.exp(99*lh)-1)*42/
dureetransition
    lf=1/(85-5)*np.log(mortalitefemme[84]/mortalitefemme[4])
    coeffdroitetransitionmortalitef=5.2349E-4*99*lf/(np.exp(99*lf)-1)*42/
dureetransition
    if not discret:
        f=lambda X,t:
(X_tplusdt(t,X,dt,fecondite,immigration,reste,survie,rencontre,logistique,K,tauxfecm
oyen,ecartype,transition,coeffdroitetransitionfecondite,coeffdroitetransitionmortali
teh,coeffdroitetransitionmortalitef,p,emmigration,mortalite,lh,lf,dureetransition)-
X)/dt
        #t,Y=Euler(f,y0,0,n,nb)
        t,Y=RK4(f,y0,0,n,nb)
        #
    else:
        nb=n
        f=lambda X,t:
(X_tplusdt(t,X,dt,fecondite,immigration,reste,survie,rencontre,logistique,K,tauxfecm
oyen,ecartype,transition,coeffdroitetransitionfecondite,coeffdroitetransitionmortali
teh,coeffdroitetransitionmortalitef,p,emmigration,mortalite,lh,lf,dureetransition))
        Y=discretfct(f,y0,n)
```

```
    yliste2=[0 for i in range(len(Y))]
    for i in range(len(Y)):
        yliste2[i]=sum([Y[i,j] for j in range(len(Y[i]))])
    #
    matrice=[[Y[len(Y)-1,i]] for i in range(len(Y[0]))]
    #

    ecartliste,ecartlistetot,arret=ecartcontinu(dt,Y,yliste2,fichierhomme,fichierfem
me,fichierpopulationinitiale,n,p,annee,nb,discret)
    return matrice,np.transpose(Y),yliste2,ecartliste,ecartlistetot,arret,nb
def enregistrement_continu():
    pass
def tableau_continu():
    pass
```

Chapitre 4

Méthode de la puissance

puissancemethode.py

```python
# -*- coding: utf-8 -*-


import numpy as np
import random as rd


from math import sqrt

def prodAx(A,x):
    n=len(A)
    y=list()
    for i in range(n):
        s=0
        for k in range(n):
            s+=A[i][k]*x[k]
        y.append(s)
    return y

def puissance(A,tol,nmax):
    n=len(A)
    y=[0]*n
    y[1]=1
    v=sum(map(lambda x1,x2: x1*x2,y,prodAx(A,y)))
    res=tol+1
    niter=1

    while (res>=tol) and (niter<=nmax):
        Ay=prodAx(A,y)
        vy=[val*v for val in y]
        Ay_vy=[val1-val2 for val1,val2 in zip(Ay,vy)]
        res=sqrt(sum(map(lambda x:x**2,Ay_vy)))
        y=prodAx(A,y)
        norm_y=sqrt(sum(map(lambda x:x**2,y)))
        y=[val/norm_y for val in y]
        v=sum(map(lambda x1,x2:x1*x2,y,prodAx(A,y)))
        niter+=1
        if niter>nmax:
            return None
    return v,y

def deflation(A,tol,niter):
    n=len(A)
    v,y=puissance(A,tol,niter)
    list_vp=[v]
    list_yp=[y]
    for compteur in range(n-1):
        A=[[A[i][j]-v*y[i]*y[j] for j in range(n)] for i in range(n)]
        v,y=puissance(A,tol,niter)
        list_vp.append(v)
        list_yp.append(y)
    return list_vp,list_yp
```

Chapitre 5

Rayon spectral de Leslie

matricelesliepositive.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 28 15:01:16 2017

@author: Utilisateur
"""
import numpy as np
from puissancemethode import *
from modelediscret import ouverturefichier,pop, mort, fec, em,
im,p,Leslie,matsurvie,matimmigration


annee = "2007"
reste = [0 for x in range(p)]

initiale,mortalite,fecondite,emmigration,immigration,fichierpopulationinitiale,femme
s,mortalitehomme,mortalitefemme,hommes,immigrationhomme,immigrationfemme,emmigration
homme,emmigrationfemme,fichierfemme,fichierhomme,ecartype = ouverturefichier(pop,
mort, fec, em, im, annee)


emmigration=[0 for i in range(len(emmigration))]
matleslie = Leslie(fecondite, matsurvie(mortalite, emmigration), reste=[0 for x in
range(p)])
matim = matimmigration(immigration)


def positive(M):
    for i in range(len(M)):
        for j in range(len(M[i])):
            if M[i][j] <= 0:
                return True

    return False

def maxmodul(a):
    m=0
    for i in a:
        if abs(i)>abs(m):
            m=i
    return m

A = matleslie
print(mortalite)
a=input("pause")
M = A
i = 1
while positive(M):
    M = np.dot(A, M)
    i += 1
    print(i)
    if i>500:
        print("calculs trop longs... la phrase qui suit est fausse")
        break

print("matrice positive à la puissance : ", i)

##########méthode de la puissance#####################
val,vect=puissance(A,1e-2,1000)
print("val : ",val,"\n","vect : ",vect)
##################eigenval#################
```

```
valeurs_propres,vecteurs_propres=np.linalg.eig(A)[0].tolist(),np.linalg.eig(A)
[1].tolist()
m=maxmodul(valeurs_propres)
print(m)
m_vecteurpropreassocie=vecteurs_propres[valeurs_propres.index(m)]
print(m_vecteurpropreassocie)
#a = input()
```

Chapitre 6

Mc Kendrick/Von Foerster

laplace.py

```
import matplotlib.pyplot as plt
from math import *

#pop=
#fert=
#mort=
pop1=[673650,673650,673650,673650,673650,644476,644476,644476,644476,644476,612407,6
12407,612407,612407,612407,593639,593639,593639,593639,593639,600867,600867,600867,6
00867,600867,612403,612403,612403,612403,612403,553200,553200,553200,553200,553200,5
00817,500817,500817,500817,500817,486172,486172,486172,486172,486172,455906,455906,4
55906,455906,455906,403243,403243,403243,403243,403243,340228,340228,340228,340228,3
40228,294105,294105,294105,294105,294105,216506,216506,216506,216506,216506,153908,1
53908,153908,153908,153908,153908,115532,115532,115532,115532,115532,71831,71831,71831,7183
1,71831,37295,37295,37295,37295,37295,13475,13475,13475,13475,13475,2917,2917,2917,2
917,434]
fert1=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,46.5,46.5,46.5,46.5,46.5,145.9,145.9,145.9,145.
9,145.9,143.8,143.8,143.8,143.8,143.8,96.7,96.7,96.7,96.7,96.7,49.9,49.9,49.9,49.9,4
9.9,16.6,16.6,16.6,16.6,16.6,4,4,4,4,4,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
mort1=[34081,34081,34081,34081,34081,3245,3245,3245,3245,3245,2839,2839,2839,2839,28
39,3524,3524,3524,3524,3524,4728,4728,4728,4728,4728,5270,5270,5270,5270,5270,5667,5
667,5667,5667,5667,6372,6372,6372,6372,6372,7451,7451,7451,7451,7451,9218,9218,9218,
9218,9218,11716,11716,11716,11716,11716,15204,15204,15204,15204,15204,19099,19099,19
099,19099,19099,22449,22449,22449,22449,22449,27172,27172,27172,27172,27172,30606,30
606,30606,30606,30606,29130,29130,29130,29130,29130,22275,22275,22275,22275,22275,11
759,11759,11759,11759,11759,4702,4702,4702,4702,4702]

pop=[pop1[i]*200 for i in range(len(pop1))]
fert=[fert1[i]/1000 for i in range(len(fert1))]
mort=[mort1[i]*200/pop[i] for i in range(len(mort1))]

fec=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.022739349,0.022739349,0.022739349,0.022739349,0.0
22739349,0.0720137614,0.0720137614,0.0720137614,0.0720137614,0.0720137614,0.06739222
93,0.0673922293,0.0673922293,0.0673922293,0.0673922293,0.0456358619,0.0456358619,0.0
456358619,0.0456358619,0.0456358619,0.0244008262,0.0244008262,0.0244008262,0.0244008
262,0.0244008262,0.0080176719,0.0080176719,0.0080176719,0.0080176719,0.0080176719,0.
0018957219,0.0018957219,0.0018957219,0.0018957219,0.0018957219]+[1/1000]*(100-49)
#print(len(fecondite))
pop=[673650,673650,673650,673650,673650,644476,644476,644476,644476,644476,612407,61
2407,612407,612407,612407,593639,593639,593639,593639,593639,600867,600867,600867,60
0867,600867,612403,612403,612403,612403,612403,553200,553200,553200,553200,553200,50
0817,500817,500817,500817,500817,486172,486172,486172,486172,486172,455906,455906,45
5906,455906,455906,403243,403243,403243,403243,403243,340228,340228,340228,340228,34
0228,294105,294105,294105,294105,294105,216506,216506,216506,216506,216506,153908,15
3908,153908,153908,153908,115532,115532,115532,115532,115532,71831,71831,71831,71831
,71831,37295,37295,37295,37295,37295,13475,13475,13475,13475,13475,2917*4/5,2917*4/5
,2917*4/5,2917*4/5,2917/5+434]
mort=[34081,34081,34081,34081,34081,3245,3245,3245,3245,3245,2839,2839,2839,2839,283
9,3524,3524,3524,3524,3524,4728,4728,4728,4728,4728,5270,5270,5270,5270,5270,5667,56
67,5667,5667,5667,6372,6372,6372,6372,6372,7451,7451,7451,7451,7451,9218,9218,9218,9
218,9218,11716,11716,11716,11716,11716,15204,15204,15204,15204,15204,19099,19099,190
99,19099,19099,22449,22449,22449,22449,22449,27172,27172,27172,27172,27172,30606,306
06,30606,30606,30606,29130,29130,29130,29130,29130,22275,22275,22275,22275,22275,117
59,11759,11759,11759,11759,0.9*2917*4/5,0.9*2917*4/5,0.9*2917*4/5,0.9*2917*4/5,2917/
5+434]
#fecondite=[fecondite[i]*2/1000 for i in range(len(fecondite))]
pop=[pop[i]*1000/(5 if i<95 else (4 if i<99 else 1)) for i in range(len(pop))]
mort=[mort[i]/5*1000/(5 if i<95 else (4 if i<99 else 1))/pop[i] for i in
range(len(mort))]
#mortalite[95:]=[0.8]*4+[1]
def itg(f,a,d):
```

```
    v=(d-a)/6
    mil=(a+d)/2
    return v*(f(a)+4*f(mil)+f(d))
def laplace(f,a,d,p,lap):
    def h(x):
        return exp(-lap*x)*f(x)
    return itg(h,a,d)
def exp_it(a):
    if a<1:
        return 1
    if a>=1 and a<=99:
        u=int(a)
        k=sum(mort[:u])
        return exp(-(k+(u-a)*mort[u]))
    else:
        return exp(-sum(mort))
def l(a):
    return fert[int(a)]*exp_it(a)
def lf(lap):
    g1=laplace(l,0,99,1,lap)
    g=1/(1-g1)
    def h(t):
        return pop[int(-t)]*exp_it(-t)*exp(-lap*t)
    def H(a):
        return l(a)*exp(-lap*a)*itg(h,-a,0)
    return g*itg(H,0,99)

print(lf(0.001))
x=[i for i in range(200)]
y=[2**(-i)*lf(2**(-i)) for i in range(200)]
plt.plot(x,y)
plt.title("Etude asymptotique de la transformée de Laplace")
plt.ylabel("x*L(f)(x)")
plt.xlabel("x")
plt.grid()
plt.legend()
plt.show()
```

Chapitre 7

Transition démographique

mortalitetransition.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun  5 15:00:05 2018

@author: Utilisateur
"""
#import tkinter as tk
#from tkinter.filedialog import askopenfilename
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.axes3d import Axes3D, get_test_data
from matplotlib import cm
import numpy as np
#from time import time
#import psutil
#pid=os.getpid()
#ps=psutil.Process(pid)
#from tkinter import messagebox
#import pygame
#from pygame.locals import *
#from time import sleep
#import os
#import random
#import gc
################################taux de mortalité################################
l=0.0841561604
coeff=-5.2349E-4*99*l/(np.exp(99*l)-1)


A=lambda t: 0.00008+coeff*t


m=lambda x,t: A(t)*np.exp(l*x)
x=[i for i in range(100)]
t=[i for i in range(42)]
z=np.array([[m(i,j) for i in x] for j in t])
X,Y=np.meshgrid(x,t)
fig = plt.figure()
ax = fig.add_subplot(111,projection="3d")
surf=ax.plot_wireframe(X,Y,z,rstride=5,cstride=5)
plt.title("Evolution des taux de mortalité en fonction de l'âge et du temps")
ax.set_xlabel("age (années)")
ax.set_ylabel("temps(années)")
ax.set_zlabel("taux de mortalité")
ax.legend()
ax.grid()
#####################################taux de fécondité#################################
variance=32.9829268103
esperance=30
A=lambda t: -8.2029E-4*t+0.1112937931034483
f=lambda x,t: A(t)*np.exp(-(x-esperance)**2/(2*variance))
x=[i for i in range(100)]
t=[i for i in range(42)]
z=np.array([[f(i,j) for i in x] for j in t])
X,Y=np.meshgrid(x,t)
fig = plt.figure()
ax = fig.add_subplot(111,projection="3d")
surf=ax.plot_wireframe(X,Y,z,rstride=5,cstride=5)
plt.title("Evolution des taux de fecondité en fonction de l'âge et du temps")
ax.set_xlabel("age (années)")
ax.set_ylabel("temps(années)")
ax.set_zlabel("taux de fecondité")
```

```
ax.legend()
ax.grid()
plt.show()
```

Chapitre 8

Paramètres gaussienne

ecart_ type.py

```
#-------------------------------------------------------------------------------
# Name:        module1
# Purpose:
#
# Author:      cmonnier
#
# Created:     22/02/2018
# Copyright:   (c) cmonnier 2018
# Licence:     <your licence>
#-------------------------------------------------------------------------------



import numpy as np
import matplotlib.pyplot as plt


def e_t(K):
    maxi=max(K)
    ind_max=K.index(maxi)
    i=ind_max
    j=ind_max
    while K[i]>maxi/2:
        i-=1
    while K[j]>maxi/2:
        j+=1
    return [i,j]



def calcul_ecart(K):
    i,j=e_t(K)
    a=K[i+1]-K[i]
    b=K[i]-a*i
    x1=1/a*(max(K)/2-b)
    a=K[j]-K[j-1]
    b=K[j]-a*j
    x2=1/a*(max(K)/2-b)
    #print(x2-x1)
    var=(x2-x1)**2/(8*np.log(2))
    return var




def gauss(K,mu,var):
    A=max(K)
    return lambda x: A*np.exp(-(x-mu)**2/(2*var))


def moy(L):
    return sum(L)/len(L)
def float_the_list(L):
    for i in range(len(L)):
        try:
            L[i]=float(L[i].replace(" ","").replace(":","0"))
        except:
            #print(len(L[i]))
            a=input("pause")
    return L
```

```
##########################calcul de la variance et de l'espérance moyenne de la
gaussienne fécondité#############################
def calculparametregaussienne(txt,ligne_debut,ligne_fin):
    fichier=open(txt,"r")
    l=fichier.readlines()
    fichier.close()
    l=[l[i].replace(",",".") for i in range(len(l))]
    l2=l[ligne_debut:ligne_fin]
    l2=[float_the_list(l2[i].split(";")[1:]) for i in range(len(l2))]
    tauxmoyen=[moy([l2[j][i] for j in range(len(l2))]) for i in range(len(l2[0]))]
    return tauxmoyen,calcul_ecart(tauxmoyen),tauxmoyen.index(max(tauxmoyen))+14
#################################################################################

taux="taux.csv"
tauxmoyen,variance,esperance=calculparametregaussienne(taux,10,39)
gaussiennefecondite=gauss(tauxmoyen,esperance,variance)
tauxmoyen=[0]*10+[tauxmoyen[0]]*5+tauxmoyen+[tauxmoyen[-1]]*(100-len(tauxmoyen)-14)


def main():
    listeecarts=[]
    for i in range(8):
        listeecarts.append(calculparametregaussienne(taux,10+i*37,39+i*37)[1])
    #print(listeecarts)
    x=np.linspace(0,100,10000)
    plt.plot(x,[max(tauxmoyen)/2 for i in range(len(x))])
    plt.plot(x,gaussiennefecondite(x),label="théorique")
    x=[i for i in range(len(tauxmoyen))]
    plt.plot(x,tauxmoyen,label="Réalité")
    plt.title("Evolution du taux moyen de fécondité en Europe en fonction de l'âge")
    plt.xlabel("classe d'âge")
    plt.ylabel("taux")
    plt.grid()
    plt.legend()
    ##########mortalite
    plt.figure()
    liste2006=[0.00403, 0.00035, 0.00021, 0.00016, 0.00013, 0.00012, 0.0001, 8e-05,
8e-05, 9e-05, 8e-05, 0.0001, 0.00012, 0.00014, 0.00016, 0.00022, 0.00027, 0.00038,
0.00043, 0.00051, 0.00052, 0.00053, 0.00051, 0.00057, 0.0006, 0.00052, 0.00061,
0.00058, 0.00061, 0.00061, 0.00061, 0.00074, 0.00073, 0.00078, 0.00088, 0.00094,
0.00108, 0.00107, 0.00125, 0.0013, 0.00145, 0.00158, 0.00176, 0.00192, 0.00213,
0.00238, 0.00277, 0.00297, 0.00323, 0.00357, 0.00401, 0.00445, 0.00451, 0.00488,
0.00508, 0.00542, 0.00589, 0.00631, 0.00645, 0.007, 0.00756, 0.00786, 0.00862,
0.0093, 0.00997, 0.00988, 0.01124, 0.01198, 0.01331, 0.01396, 0.01514, 0.01688,
0.01848, 0.02039, 0.02249, 0.02478, 0.02799, 0.03062, 0.03444, 0.03807, 0.04294,
0.04782, 0.05415, 0.06093, 0.06932, 0.13735, 0.13735, 0.13735, 0.13735, 0.13735,
0.13735, 0.13735, 0.13735, 0.13735, 0.13735, 0.13735, 0.13735, 0.13735, 0.13735, 1]
    listereel=[0.00387, 0.00033, 0.00021, 0.00015, 0.00012, 9e-05, 9e-05, 9e-05,
0.0001, 8e-05, 7e-05, 9e-05, 0.00011, 0.00011, 0.00015, 0.0002, 0.00025, 0.00033,
0.00044, 0.00044, 0.00054, 0.00054, 0.00051, 0.00058, 0.00054, 0.00058, 0.00063,
0.00062, 0.00061, 0.0006, 0.00063, 0.00063, 0.00072, 0.00073, 0.0008, 0.00088,
0.00099, 0.00111, 0.00117, 0.00121, 0.00136, 0.00155, 0.00173, 0.00181, 0.00208,
0.00238, 0.00252, 0.00279, 0.00309, 0.00356, 0.00373, 0.00409, 0.00448, 0.0048,
0.00506, 0.00548, 0.00567, 0.00631, 0.00658, 0.00688, 0.00756, 0.00776, 0.00849,
0.00924, 0.00961, 0.01034, 0.01041, 0.01185, 0.01308, 0.01378, 0.01515, 0.01617,
0.01801, 0.01954, 0.02166, 0.02404, 0.02679, 0.0296, 0.03375, 0.03773, 0.04258,
0.04723, 0.05326, 0.05969, 0.06793, 0.13373, 0.13373, 0.13373, 0.13373, 0.13373,
0.13373, 0.13373, 0.13373, 0.13373, 0.13373, 0.13373, 0.13373, 0.13373, 0.13373,1]
    listereel2=[0.00397, 0.00033, 0.00019, 0.00018, 0.00013, 0.0001, 0.0001, 7e-05,
```

```
8e-05, 8e-05, 9e-05, 0.0001, 0.0001, 0.0001, 0.00014, 0.00018, 0.00025, 0.00028,
0.00043, 0.00045, 0.00047, 0.00053, 0.00056, 0.00055, 0.00057, 0.00058, 0.00057,
0.0006, 0.00059, 0.00062, 0.00063, 0.00064, 0.0007, 0.00073, 0.00083, 0.00086,
0.00098, 0.00103, 0.00113, 0.00128, 0.00136, 0.00152, 0.00165, 0.00183, 0.00208,
0.00218, 0.0025, 0.00275, 0.00306, 0.00345, 0.00368, 0.00414, 0.00442, 0.0047,
0.00495, 0.00552, 0.00588, 0.006, 0.0065, 0.00675, 0.00728, 0.00808, 0.00844,
0.00895, 0.00973, 0.01028, 0.01112, 0.0109, 0.01279, 0.01356, 0.01465, 0.01632,
0.01772, 0.01956, 0.02155, 0.02415, 0.0265, 0.02986, 0.0328, 0.03748, 0.0418,
0.04751, 0.05334, 0.06043, 0.06787, 0.13248, 0.13248, 0.13248, 0.13248, 0.13248,
0.13248, 0.13248, 0.13248, 0.13248, 0.13248, 0.13248, 0.13248, 0.13248, 0.13248, 1]
    x=np.linspace(0,87,10000)
    l=1/(85-5)*np.log(listereel[84]/listereel[4])
    #coeffdroitetransitionmortalite=5.2349E-4*99*l/(np.exp(99*l)-1)
    f=lambda x: listereel[4]*np.exp(l*(x-4))
    plt.plot(x,f(x),label="théorique")
    x=[i for i in range(86)]
    plt.plot(x,listereel2[:86],label="2008")
    plt.plot(x,listereel[:86],label="2007")
    plt.plot(x,liste2006[:86],label="2006")
    plt.title("Evolution du taux de mortalité en France en fonction de l'âge 2006 à
2008")
    plt.xlabel("classe d'âge")
    plt.ylabel("taux")
    plt.grid()
    plt.legend()
    plt.show()

if __name__ == '__main__':
    main()
```