L'épidémiologie mathématique

Marouane IBN BRAHIM
Numéro de dossier : 1313

Juillet 2019

Table des matières

Première partie

Introduction

L'épidémiologie daterait du Ve siècle avant J.C. Ainsi, le médecin grec Hippocrate (460 av. J.-C,377 av. J.-C) serait le premier épidémiologue de l'histoire. Cependant, les mathématiques n'avaient pas leur place en épidémiologie, qui était une discipline relevant exclusivement de la médecine.

Ce n'est qu'au $XVIII^e$ siècle avec les travaux de Daniel Bernoulli (1700-1782) que l'on commence à comprendre leur importance, et qu'au XX^e siècle qu'elle commence réellement à se développer telle que nous la connaissons aujourd'hui, grâce aux travaux de W.O. Kermack et A.G. McKendrick.

L'épidémiologie mathématique est une discipline scientifique relevant des mathématiques appliquées, qui consiste à étudier le transport d'une épidémie au sein d'une population. De manière déterministe, elle consiste en :

— Un paramètrage ;

— Une mise en équations ;

— La résolution des équations.

Il suffit ensuite de comparer les résultats obtenus aux statistiques réelles pour juger de l'efficacité du modèle.

Deuxième partie

Les modèles compartimentaux

1 Définition

Les modèles compartimentaux représentent une manière assez fréquente de modéliser le transport d'une épidémie au sein d'une population. Ils consistent à diviser la population en plusieurs compartiments, avant de décrire les interactions entre eux, par exemple à travers un système d'équations différentielles.

Quand nous aurons déterminé ces équations on pourra les résoudre numériquement.

2 Quelques notations que nous utiliserons dans toute la suite

— N désigne la population totale à un instant t

— S désigne le nombre de susceptibles à un instant t

— I désigne le nombre d'infectieux à un instant t

— E désigne le nombre d'éxposés à un instant t

— R désigne le nombre de personnes guéries et immunisées à un instant t

— X' désigne la dérivée d'une grandeur X par rapport au temps

— β désigne le taux de contagion

— γ désigne le taux de guérison

— μ désigne le taux de létalité

3 Exemples de modèles

3.1 Le modèle SIS

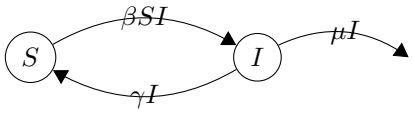
Le modèle SIS (pour Susceptible-Infectieux-Susceptible) décrit généralement une épidémie contre laquelle les guéris ne sont pas immunisés.

Paramétrage : Ce modèle consiste à diviser la population en deux compartiments :

— Susceptibles : Ce sont les individus non infectées, susceptibles d'attraper la maladie ;

— Infectieux : Ce sont les individus infectés et contagieux.

Mise en équation : Commençons par un schéma pour comprendre concrètement les interactions entre les compartiments :



Le modèle SIS

— Le nombre βSI représente le nombre de personnes qui tombent malades par unité de temps.

— Le nombre de personnes guéries (resp. mortes) par unité de temps est proportionnel à I. On le note γI (resp. μI).

On déduit le système d'équations suivant :

$$\begin{cases} S' = -\beta SI + \gamma I \quad (1) \\ I' = \beta SI - \gamma I - \mu I \quad (2) \end{cases}$$

3.2 Le modèle SIR

Le modèle SIR (pour Susceptible-Infectieux-Retiré) modélise en général une épidémie contre laquelle les guéris, contrairement à celle du modèle précedent, sont immunisés.

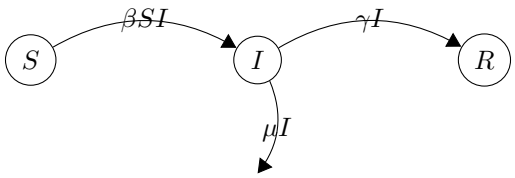
Paramétrage : Ce modèle consiste à diviser la population en trois compartiments :

— Susceptibles : Ce sont justement les individus non infectés.

— Infectieux : Ce sont les individus infectés et contagieux

— Retirés : Ce sont les individus guéris (et immunisés) contre la maladie

Mise en équation : Le schéma pour ce modèle est :



Le modèle SIR

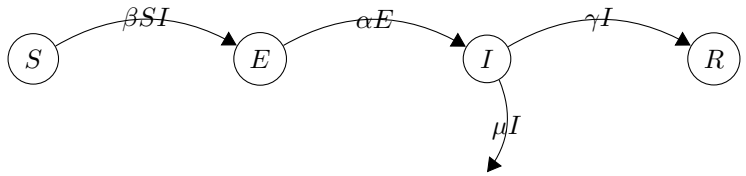
d'où le système différentiel suivant :

$$\begin{cases} S' = -\beta SI \quad (3) \\ I' = \beta SI - \gamma I - \mu I \quad (4) \\ R' = \gamma I \quad (5) \end{cases}$$

3.3 Le modèle SEIR

Certaines épidémies se caractérisent par une période d'incubation. C'est une période durant laquelle un individu est infecté, sans présenter de symptôme ni être contagieux (Infectieux). Pour le prendre en compte, on ajoute le compartiment des exposés (E) au modèle précedent .

On obtient ainsi le modèle SEIR (Susceptibles-Exposés-Infectieux-Retiré), décrit par le schéma suivant :



Le modèle SEIR

d'où le système différentiel suivant :

$$\begin{cases} S' = -\beta SI \quad (6) \\ E' = \beta SI - \alpha E \quad (7) \\ I' = \alpha E - \mu I - \gamma I \quad (8) \\ R' = \gamma I \quad (9) \end{cases}$$

4 Exemple : Le virus Ebola

Le virus Ebola a été découvert pour la première fois durant les années 70. Depuis, plusieurs épidémies se sont succédées, notamment l'épidémie qui a ravagé plusieurs pays d'Afrique entre 2013 et 2016. En Sierra Leone, ce virus a touché plus de 14.000 cas suspectés et a fait plus de 3900 morts en 2 ans.
On se propose de faire l'étude de son transport au sein de la population de Sierra Leone.

Etant donné les caractéristiques de cette épidémie, il semble plus adéquat de décrire son transport par les modèles SIR et SEIR. En effet, tout guéri de cette maladie est immunisé, et elle présente une période d'incubation qui varie de 2 jours à 21 jours.

Problématique : Comment peut on déterminer les valeurs des constantes α, β, γ et μ qui correspondent à notre modèle et au transport étudié ?

Nous allons voir dans la partie suivante deux méthodes potentielles pour résoudre ce problème et conclure quant à leur efficacité.

Troisième partie

Détermination des constantes

5 Méthode d'interpolation

Théorème 1 (Les polynômes d'interpolation de Lagrange)**.** *Etant donné un entier non nul n et deux familles : $(x_i)_{1\leq i\leq n+1}$ ordonnée par ordre croissant et $(y_i)_{1\leq i\leq n+1}$, il existe un unique polynôme P de degré inférieur ou égal à n tel que :*

$$\forall i \in [\![1\ ;\ n+1]\!] \quad , \quad P(x_i) = y_i$$

En effet l'existence est assurée par le polynôme P donné par :

$$L_i(X) = \prod_{j=1, j\neq i}^{n+1} \frac{X - x_j}{x_i - x_j} \quad \text{et} \quad P(X) = \sum_{i=1}^{n+1} y_i L_i(X)$$

Cas pratique : On récupère les statistiques officielles de l'OMS et on détermine les polynômes qui interpolent S et I aux points du temps, qu'on note encore S et I. Pour SIR, l'équation (3) donne : $\beta = -\frac{S'}{SI}$. On calcule une moyenne de cette quantité.

Etant donné le nombre important d'opérations pour calculer le polynôme interpolateur, l'erreur cumulée par la machine est importante, on diminue donc le nombre de points on ne prenant que ceux espacés au moins d'un certain i.

— Pour $i = 25$ jours on obtient : $\beta = 1.505 \quad . \quad 10^{-8}$

— Pour $i = 30$ jours on obtient : $\beta = 5.556 \quad . \quad 10^{-8}$

On déduit donc que cette méthode est trop instable et qu'elle est donc à rejeter.

6 Méthode des moindres carrés

6.1 Présentation de la méthode

Pour remédier au problème de l'erreur cumulée, on essaie de diminuer le degré du polynôme (le choix du degré 2 est retenu), et ainsi chercher le polynôme le plus proche, au sens de la distance usuelle entre fonctions sur un intervalle $[a, b]$.

Théorème 2. *Etant donné un entier $n > 2$ et deux familles : $(x_i)_{1\leq i\leq n+1}$ ordonnée par ordre croissant et $(y_i)_{1\leq i\leq n+1}$, le polynôme de degré 2 : $P(X) = a + bX + cX^2$ qui minimise la quantité $\sum_{i=1}^{n+1}(y_i - P(x_i))^2$ existe et vérifie :*

$$^tAAX = {}^tAB$$

où

$$A = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ 1 & x_{n+1} & x_{n+1}^2 \end{pmatrix} \quad X = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \quad B = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{n+1} \end{pmatrix}$$

Cas pratique : En inversant (numériquement) ^tAA, on détermine les coefficients du polynôme.
Dans toute la suite, on suppose que $\mu = 0$.
On cherche le polynôme précedent pour S, I et E.
Pour SIR, on détermine : β par $\beta = -\frac{S'}{SI}$, et γ par $\gamma = -\frac{I' + S'}{I}$.
Pour SEIR, on détermine β par $\beta = -\frac{S'}{SI}$, α par $\alpha = -\frac{E' + S'}{E}$, et γ par $\gamma = -\frac{S' + E' + I'}{I}$.

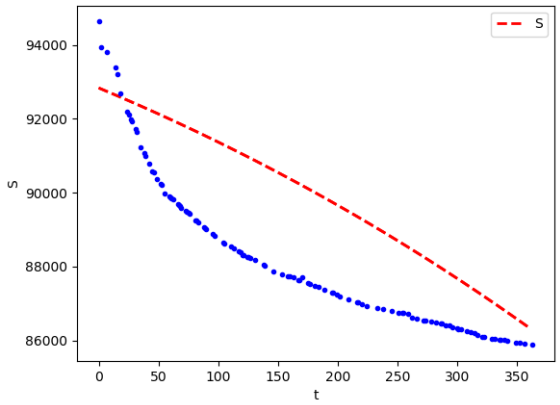
6.2 Résultats

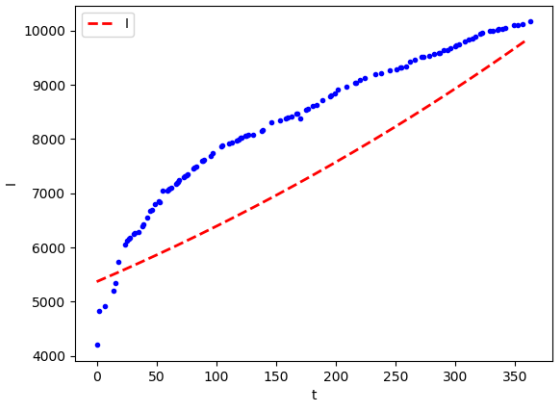
Pour le modèle SIR : En implémentant la méthode des moindres carrés en langage Python, on obtient :

— $\beta = 2.714 \quad . \quad 10^{-8}$

— $\gamma = 7.57 \quad . \quad 10^{-4}$

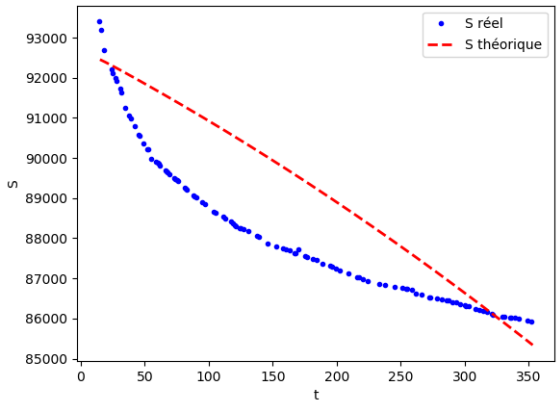
Les figures suivantes représentent les courbes réelles (en bleu) et théoriques (en rouge) pour le compartiment I (à droite) et S (à gauche).

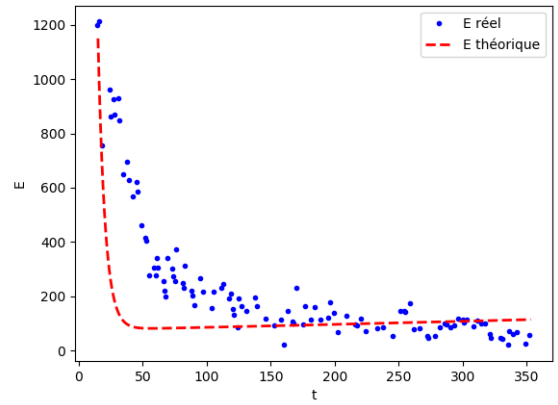


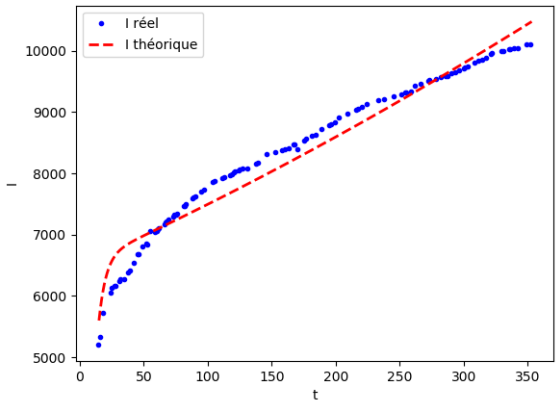


Pour le modèle SEIR : On obtient de la même manière :

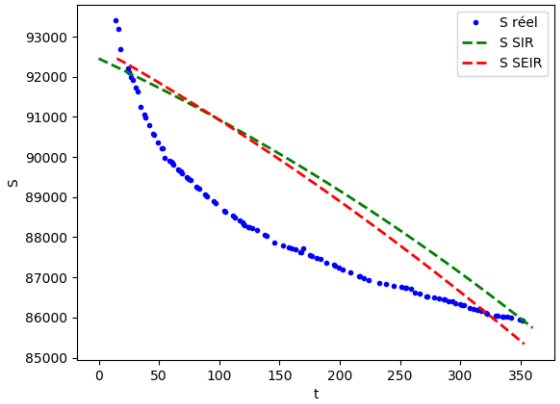
- $\beta = 2.806 \quad . \quad 10^{-8}$
- $\gamma = 1.139 \quad . \quad 10^{-3}$
- $\alpha = 0.1775$

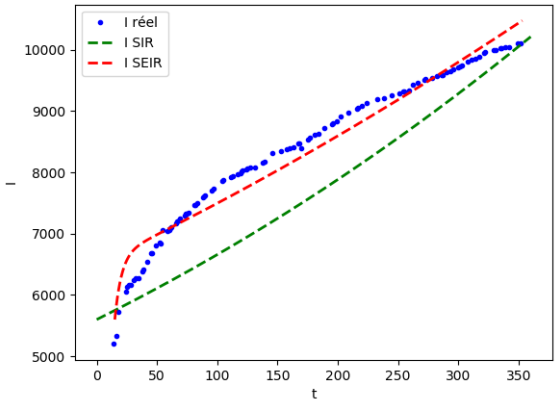






Les courbes suivantes représentent la superposition des résultats des deux méthodes :





Quatrième partie

Annexe

7 Listings des codes utilisés

Liste contenant les enregistrements de l'OMS [date, Nombre de cas, Nombre de morts] :

```
data=[(0,5368,1169),(2,6073,1250),(7,6190,1267),(14,6599,1398),(16,6802,1463),(18,7312,1583),
(24,7798,1742),(25,7897,1768),(27,8014,1857),(28,8069,1899),(31,8273,2033),(32,8356,2085),
(35,8759,2477),(38,8939,2556),(39,9004,2582),(42,9203,2655),(45,9409,2732),(46,9446,2758),
(49,9633,2827),(52,9772,2915),(53,9780,2943),(55,10030,2977),(59,10094,3049),(60,10124,3062),
(61,10150,3067),(62,10186,3083),(66,10306,3132),(67,10340,3145),(68,10362,3153),(69,10400,3159),
(73,10491,3195),(74,10518,3199),(75,10537,3207),(76,10561,3216),(81,10740,3276),(82,10756,3286),
(83,10792,3301),(88,10934,3341),(89,10954,3350),(90,10987,3363),(95,11103,3408),(97,11155,3423),
(104,11341,3479),(105,11370,3490),(111,11466,3546),(113,11497,3565),(117,11586,3611),
(119,11619,3629),(120,11677,3655),(121,11696,3663),(124,11742,3687),(125,11751,3691),
(127,11779,3702),(131,11829,37),(138,11943,3792),(139,11974,3799),(146,12139,3832),(153,12201,3857),
(158,12256,3872),(160,12267,3877),(163,12294,3885),(167,12362,3895),(168,12371,3899),(170,12287,3901),
(175,12440,3903),(177,12470,3904),(181,12519,3904),(184,12536,3904),(189,12632,3907),
(195,12696,3908),(196,12706,3908),(199,12745,3911),(202,12816,3911),(209,12884,3913),
(216,12962,3919),(217,12965,3919),(220,13012,3926),(224,13059,3928),(233,13129,3933),(238,13155,3940),
(245,13209,3947),(251,13241,3949),(254,13262,3949),(255,13264,3949),(259,13290,3951),(262,13387,3951),
(266,13406,3951),(272,13465,3951),(273,13470,3951),(278,13489,3952),(282,13518,3952),(286,13538,3952),
(287,13541,3952),(290,13586,3952),(293,13603,3953),(296,13638,3953),(300,13670,3953),(301,13683,3953),
(303,13697,3953),(308,13756,3953),(311,13785,3955),(314,13811,3955),(317,13846,3955),
(321,13894,3955),(322,13911,3955),(329,13945,3955),(331,13956,3955),(335,13978,3955),
(336,13982,3955),(339,13992,3955),(342,13999,3955),(349,14052,3955),(352,14066,3955),(356,14078,3955),
(363,14122,3955)]
```

Résolution des systèmes différentiels non linéaires

```
import numpy as np
from scipy.integrate import odeint
```

```
import matplotlib.pyplot as plt

#POUR SIR
def f(s,t,b,G,alpha):
    S=s[0]
    I=s[1]
    R=s[2]
    dSdt=-b*S*I
    dIdt=b*S*I - (G+alpha)*I
    dRdt=G*I
    return [dSdt,dIdt,dRdt]

t=np.linspace(0,360,1000000)
s0=[92837,5368,0]

s=odeint(f,s0,t, (2.7141037173344333e-08,0.0007573689040734204,0))

plt.plot(t,s[:,0],'r--', linewidth=2.0)
plt.plot(t,s[:,1],'r--', linewidth=2.0)
plt.plot(t,s[:,2],'b.', linewidth=2.0)
plt.xlabel("t")
plt.ylabel("S")
plt.legend(["S"])
plt.show()

#POUR SEIR
def f(s,t,b,G,alpha,nu):
    S=s[0]
    E=s[1]
    I=s[2]
    R=s[3]
    dSdt=-b*S*I
    dEdt=b*S*I-alpha*E
    dIdt=alpha*E - (G+nu)*I
    dRdt=G*I
    return [dSdt,dEdt,dIdt,dRdt]

t=np.linspace(0,353,100000)
s0=[92837,822,4199,0]

s=odeint(f,s0,t,(3.049348123160306e-08,0.0011394041241835893,0.17750651160832748,0))

plt.plot(t,s[:,0],'r--', linewidth=2.0)
plt.plot(t,s[:,1],'b-', linewidth=2.0)
plt.plot(t,s[:,2],'b', linewidth=2.0)
plt.plot(t,s[:,3],'y.', linewidth=2.0)
plt.xlabel("t")
plt.ylabel("S[S,I]")
plt.legend(["S","I"])
```

```
plt.show()
```

Construction de la R-algèbre des polynômes

```
# Construction de la R-algèbre des polynômes
def normal(P):
    try:
        R=[]
        i=len(P)-1
        while P[i]==0:
            i-=1
        P=P[:i+1]
        return P
    except:  return [0]

def multi(k,P):
    return normal([k*a for a in P])

def somme(P,Q):
    P=normal(P)
    Q=normal(Q)
    if len(P)!=len(Q):
        if len(P)>len(Q):
            while len(Q)!=len(P):
                Q.append(0)
        else:
            while len(Q)!=len(P):
                P.append(0)
    R=[]
    for i in range(len(P)):
        R.append(P[i]+Q[i])
    return normal(R)

def Cauchy(P,Q):
    p=len(P)
    s=0
    for i in range(p):
        s+=P[i]*Q[p-i-1]
    return s


def produit(P,Q):
    P=normal(P)
    Q=normal(Q)
    if len(P)!=len(Q):
        if len(P)>len(Q):
            while len(Q)!=len(P):
                Q.append(0)
        else:
            while len(Q)!=len(P):
```

```
                P.append(0)
    p=len(P)
    R=[]
    for i in range(p):
        R.append(Cauchy(P[:i+1],Q[:i+1]))
    for i in range(p-1):
        R.append(Cauchy(P[i+1:],Q[i+1:]))
    return normal(R)

def Derivation(P):
    p=len(P)
    if p==1:
        return [0]
    else:
        return [i*P[i] for i in range(1,p)]

def valeur(P,t):
    s=0
    for i in range(len(P)):
        s+=P[i]*(t**i)
    return s
```

Méthode d'interpolation de Lagrange

```
# Définition des polynômes de Lagrange et fcts associées
def Li(X,i):
    P=[1]
    for j in range(len(X)):
        if j!=i:
            a=-X[j]/(X[i]-X[j])
            b=1/(X[i]-X[j])
            P=produit(P,[a,b])
    return normal(P)

def Lagrange(X,Y):
    u=len(X)
    R=[0]
    for i in range(u):
        R=somme(R,multi(Y[i],Li(X,i)))
    return normal(R)

def poltofct(X,Y):
    def g(t):
        return valeur(Lagrange(X,Y),t)
    return g
#reduction nbr de points
temps=[i[0] for i in data]

liste=[(0,5368,1169)]
temps2=[0]
```

```
27
28  l=len(data)
29  for i in range(l):
30      if data[i][0]-liste[-1][0]>=30:
31          liste.append(data[i])
32          temps2.append(temps[i])
33
34  data = liste
35  temps=temps2
36
37  #Operateurs sur les fonctions
38  def integrateur(f,a, b, n):
39      integral=0
40      for k in range(n+1):
41          integral+=f(k*(b-a)/n)*(b-a)/n
42      return integral
43
44  def moyenne(f,a,b):
45      return integrateur(f,a,b,10000)/(b-a)
46
47  #Calcul des constantes par interpolartion
48  S0=int(input("Nombre initial de susceptibles : "))
49  S=[S0-i[1] for i in data]
50  I=[i[1]-i[2] for i in data]
51
52  s=poltofct(temps,S)
53  i=poltofct(temps,I)
54  dsdt=lambda t : derivative(s,t)
55
56  beta = lambda t: -dsdt(t)/(s(t)*i(t))
57  beta= moyenne(beta,0,363)
```

Méthode des moindres carrés pour SIR

```
1   #Methode des moindres carres pour SIR
2   S0=int(input("Nombre initial de susceptibles : "))
3   temps=[i[0] for i in data]
4   S=[S0-i[1] for i in data]
5   I=[i[1]-i[2] for i in data]
6
7   plt.plot(temps,S,'b.')
8   plt.plot(temps,I,'b.')
9   plt.legend(['S'])
10
11  A=[]
12  BS=[]
13  BI=[]
14
15  for i in temps:
16      A.append([1,i,i*i])
```

```
A=np.array(A)
tA=A.transpose()

for j in S:
    BS.append([j])
for j in I:
    BI.append([j])

BS=np.array(BS)
BI=np.array(BI)

XS=np.dot((A.transpose()),A)
XS=np.linalg.inv(XS)
XS=np.dot(XS,tA)
XS=np.dot(XS,BS)
XS=list(XS[0])+list(XS[1])+list(XS[2])
dXSdt=Derivation(XS)

XI=np.dot((A.transpose()),A)
XI=np.linalg.inv(XI)
XI=np.dot(XI,tA)
XI=np.dot(XI,BI)
XI=list(XI[0])+list(XI[1])+list(XI[2])
dXIdt=Derivation(XI)

s = lambda t : valeur(XS,t)
i = lambda t : valeur(XI,t)
dsdt = lambda t : valeur(dXSdt,t)
didt= lambda t : valeur(dXIdt, t)

beta = lambda t : -(dsdt(t))/(s(t)*i(t))
beta = moyenne(beta,0,363)

gamma= lambda t : beta*s(t)-(didt(t)/i(t))
gamma = moyenne(gamma,0, 363)
```

Méthode des moindres carrés pour SEIR

```
#Methode des moindres carres pour SEIR
def plusproche10(d,l):
    d+=10
    mini=0
    while l[mini+1][0]<=d:
        mini+=1
    if l[mini+1][0]-d<=d-l[mini][0]:
        return mini+1
    else:
        return mini

#algo:
```

```
13
14  S0=int(input("Nombre initial de susceptibles : "))
15  temps=[i[0] for i in data if i[0]<=353]
16  S=[S0-i[1] for i in data if i[0]<=353]
17  I=[i[1]-i[2] for i in data if i[0]<=353]
18  E=[]
19  i=0
20  while data[i][0]<=353:
21      E.append(data[plusproche10(data[i][0],data)][1]-data[i][1])
22      i+=1
23
24  plt.plot(temps,S,'b.')
25  plt.plot(temps,I,'b.')
26  plt.plot(temps,E,'b.')
27  plt.legend(['S'])
28
29  A=[]
30  BS=[]
31  BI=[]
32  BE=[]
33
34  for i in temps:
35      A.append([1,i,i*i])
36  A=np.array(A)
37  tA=A.transpose()
38
39  for j in S:
40      BS.append([j])
41  for j in I:
42      BI.append([j])
43  for j in E:
44      BE.append([j])
45
46  BS=np.array(BS)
47  BI=np.array(BI)
48  BE=np.array(BE)
49
50  XS=np.dot((A.transpose()),A)
51  XS=np.linalg.inv(XS)
52  XS=np.dot(XS,tA)
53  XS=np.dot(XS,BS)
54  XS=list(XS[0])+list(XS[1])+list(XS[2])
55  dXSdt=Derivation(XS)
56
57  XI=np.dot((A.transpose()),A)
58  XI=np.linalg.inv(XI)
59  XI=np.dot(XI,tA)
60  XI=np.dot(XI,BI)
61  XI=list(XI[0])+list(XI[1])+list(XI[2])
```

```
dXIdt=Derivation(XI)

XE=np.dot((A.transpose()),A)
XE=np.linalg.inv(XE)
XE=np.dot(XE,tA)
XE=np.dot(XE,BE)
XE=list(XE[0])+list(XE[1])+list(XE[2])
dXEdt=Derivation(XE)

s = lambda t : valeur(XS,t)
i = lambda t : valeur(XI,t)
e = lambda t : valeur(XE,t)
dsdt = lambda t : valeur(dXSdt,t)
didt= lambda t : valeur(dXIdt, t)
dedt= lambda t : valeur(dXEdt, t)
beta = lambda t : -(dsdt(t))/(s(t)*i(t))
alpha= lambda t : -(dsdt(t)+dedt(t))/e(t)
gamma= lambda t: -(dsdt(t)+dedt(t)+didt(t))/i(t)
beta = moyenne(beta,0,353)
alpha=moyenne(alpha,0,353)
gamma=moyenne(gamma,0,353)
```

8 Preuves de certains résultats

8.1 Preuve du théorème 1

L'existence est assurée par le polynôme P.

Unicité : Si un second polynôme Q vérifiant les mêmes hypothèses existe, alors il coïncide avec le polynôme P en $n+1$ points donc le polynôme $P-Q$ s'annule $n+1$ fois. Or $P-Q$ est la différence de deux polynômes de degrés inférieurs ou égaux à n donc son degré est au maximum n, donc c'est le polynôme nul.

8.2 Preuve du théorème 2

Soit $P = a + bX + cX^2 \in \mathbb{R}_2[X]$. On a :

$$\begin{aligned}\sum_{i=1}^{n+1}(y_i - P(x_i))^2 &= \sum_{i=1}^{n+1}(y_i - a - bx_i - cx_i^2)^2 \\ &= \|AX - B\|\end{aligned}$$

où

$$A = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ 1 & x_{n+1} & x_{n+1}^2 \end{pmatrix} \quad X = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \quad B = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{n+1} \end{pmatrix}$$

La matrice A est de rang 3. En effet si $X = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}$ est solution de $AX = 0$ alors $x_1, x_2, ..., x_{n+1}$ sont racines du polynôme : $Q(X) = \alpha_0 + \alpha_1 X + \alpha_2 X^2$. Dans le cadre de notre étude $n+1 > 2$ donc c'est le polynôme nul,

d'où le résultat.

Soit C le projeté orthogonal de B sur $Im(A)$. A étant de rang 3.
L'application $X \mapsto AX$ de $\mathcal{M}_{3,1}(\mathbb{R})$ vers $Im(A)$ est injective donc il existe un unique X_0 pour lequel le minimum est atteint. On a :

$$\begin{array}{rcc} AX_0 = p_{Im(A)}(B) & \iff & \forall X \in \mathcal{M}_{3,1}(\mathbb{R}) : AX_0 - B \perp AX \\ & \iff & \forall X \in \mathcal{M}_{3,1}(\mathbb{R}) : {}^t(AX)(AX_0 - B) = 0 \\ & \iff & \forall X \in \mathcal{M}_{3,1}(\mathbb{R}) : {}^tX({}^tAAX_0 - {}^tAB) = 0 \\ & \iff & {}^tAAX_0 = {}^tAB \end{array}$$

Références

[1] Alain-Jacques VALLERON : La Jaune et La Rouge, Brève histoire de l'épidémiologie avant le XXe siècle : *https://www.lajauneetlarouge.com/article/breve-histoire-de-lepidemiologie-avant-lexxe-siecle#.XDCk9MYo9uQ,*

[2] Maia Martcheva : *An Introduction to Mathematical Epidemiology* : Springer, 2015

[3] Fred Brauer, Carlos Castillo-Chàvez : *Mathematical Models in Population Biology and Epidemiology* : Springer, 2001

[4] Paul Benkimoun : Le Monde, 1976, à la découverte du virus Ebola : *https://www.lemonde.fr/planete/article/2014/08/10/1976-a-la-decouverte-du-virusebola_4469215_3244.html* , Publié le 10 août 2014, Mis à jour le 11 août 2014, Consulté le 20 décembre 2018

[5] Organisation Mondiale de la Santé : Ebola data and statistics : *http://apps.who.int/gho/data/view.ebola-sitrep.ebola-summary-latest ?lang=en*

[6] Gloria Faccanoni : Analyse numérique. Recueil d'exercices corrigés et aide-mémoire : *http ://faccanoni.univ-tln.fr/user/enseignements/20142015/M33_L2.pdf*