Modélisation et Contrôle de maladies en ville

Modèle des réseaux complexes stochastiques de contact - Contrôle de la propagation de maladies infectieuses en ville

Younes-Jihad Boumoussou

Juillet 2023

Introduction

Plan de l'étude

Définition

Transmission d'une maladie infectieuse

Modélisation mathématique par réseaux complexes

Réseaux complexes de contact

Exemple de réseau complexe

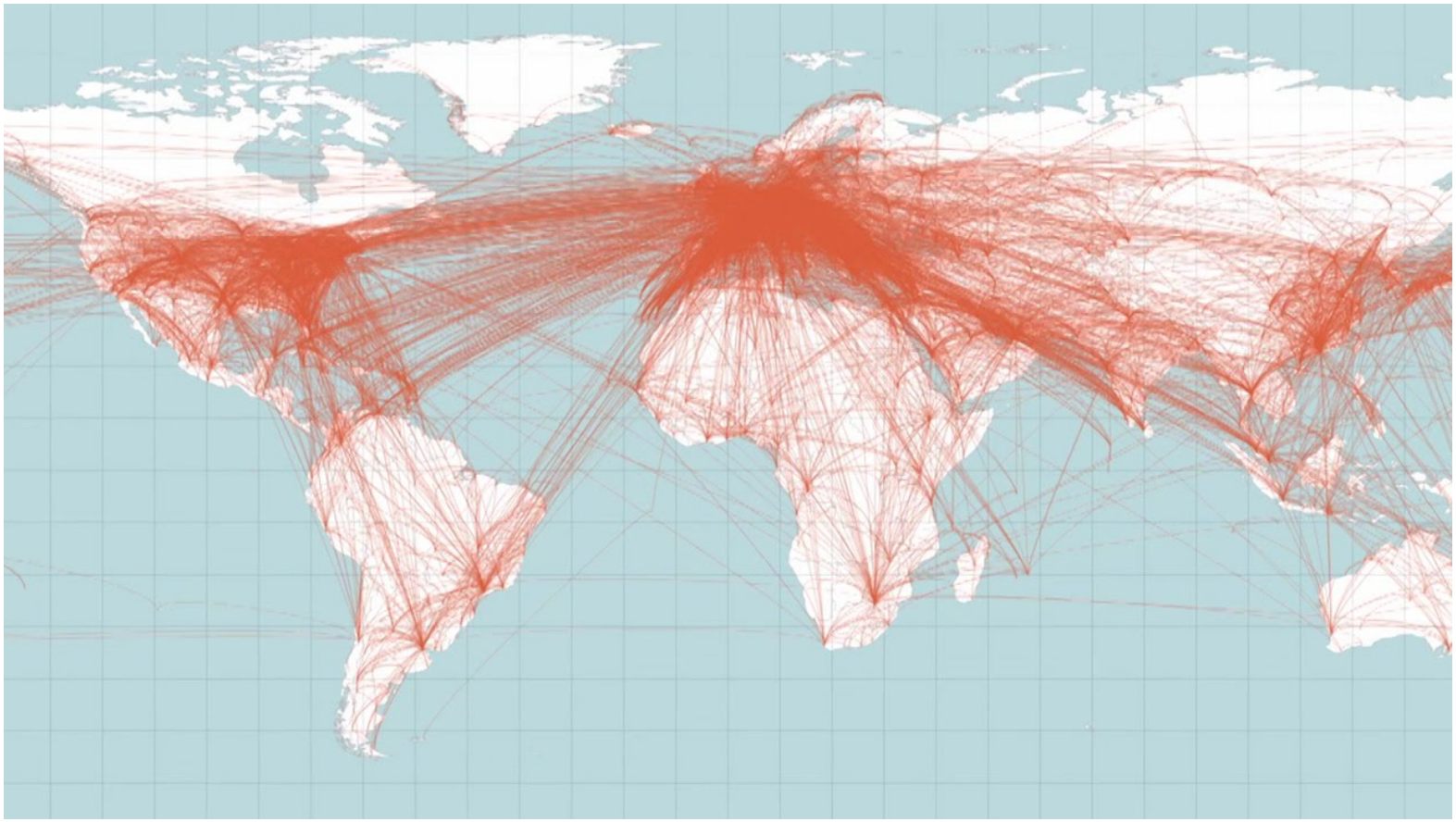
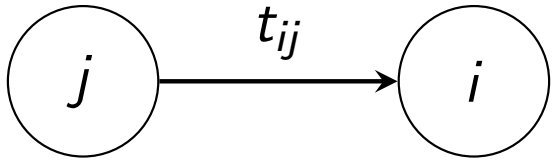


Figure: Réseau complexe mondial

Etant donné une population $\Omega = \{1, ..., n\}$,
Soit une paire $(i, j) \in \Omega^2$, j étant infecté, i étant sain et soit $t_{ij}\delta t$ la probabilité que j infecte i pendant une durée δt.



Soit $T_{ij}(t)$ la probabilité que i soit infecté par j à l'instant t.

$$1 - T_{ij}(t) = \lim_{\delta t \to 0} (1 - t_{ij}\delta t)^{t/\delta t} = e^{-t_{ij}t}$$

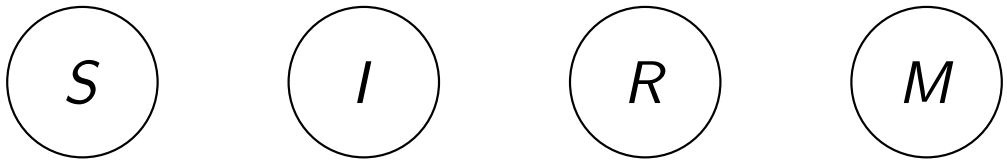
$$\implies T_{ij}(t) = 1 - e^{-t_{ij}t}$$

On définit alors :

$$T(t) = \begin{pmatrix} T_{11}(t) & T_{12}(t) & \cdots & T_{1n}(t) \\ T_{21}(t) & T_{22}(t) & \cdots & T_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ T_{n1}(t) & T_{n2}(t) & \cdots & T_{nn}(t) \end{pmatrix}$$

Que l'on nomme la matrice de transmission à l'instant t.

On définit 4 états pour les individus de la population :



- Infectés à un instant donné de la propagation, on note leur ensemble I
- Sains, on note leur ensemble S
- Rétablis, on note leur ensemble R
- Décédés, on note leur ensemble M

- Soit $I_k(t), M_k(t), R_k(t)$ les variables aléatoires respectives qui reçoivent 1 si l'individu k est infecté à un instant donné de la propagation, décédé ou rétabli à l'instant t et 0 sinon.
- Soit $I(t), M(t), R(t)$ les variables aléatoires respectives qui reçoivent le nombre total d'individus infectés à un instant donné de la propagation, décédés et rétablis sur l'ensemble de la population à l'instant t.

Et on note, pour simplifier :

$$p_k(t) = \mathbb{P}(I_k(t) = 1), q_k(t) = \mathbb{P}(M_k(t) = 1), r_k(t) = \mathbb{P}(R_k(t) = 1)$$

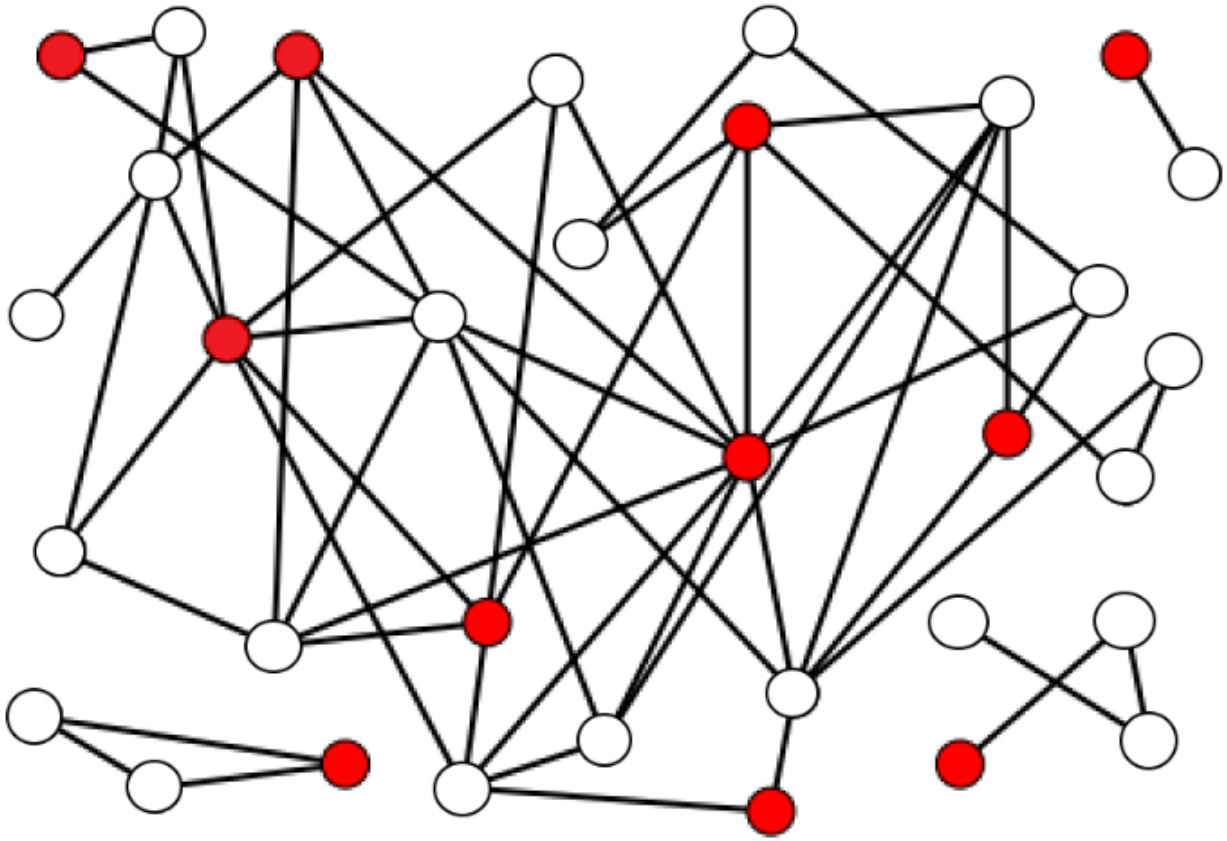
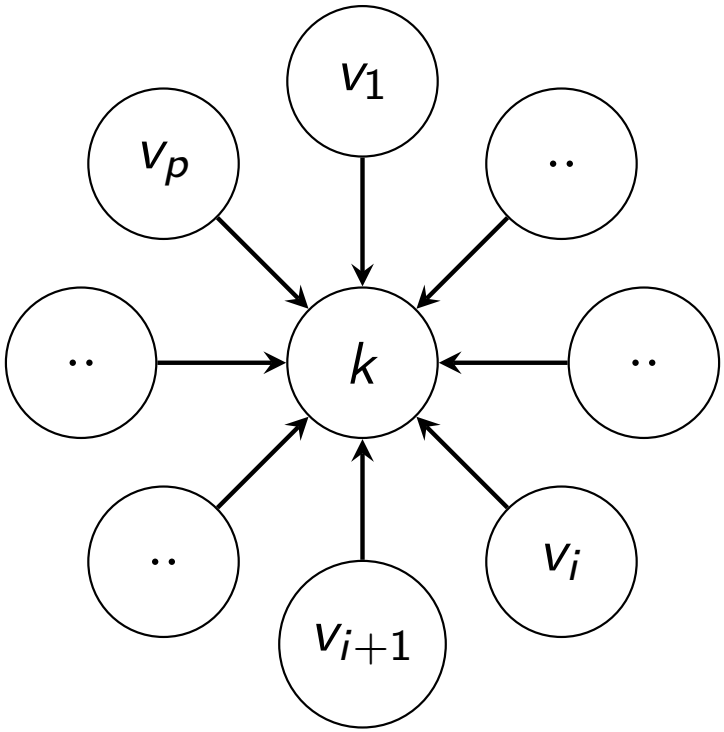


Figure: Un réseau complexe de contact contenant des sains et des infectés

En notant $voisins(k) = \{i \in \Omega, \exists t \geq 0, T_{ki}(t) > 0\} = \{v_i, 1 \leq i \leq p\}$ les voisins de k.



Alors $\forall k \in \Omega$,

$$p_k(t + dt) = \sum_{i \in voisins(k)} p_i(t) T_{ki}(t)$$

D'où :

$$\begin{pmatrix} p_1(t+dt) \\ p_2(t+dt) \\ \vdots \\ p_n(t+dt) \end{pmatrix} = \begin{pmatrix} T_{11}(t) & T_{12}(t) & \cdots & T_{1n}(t) \\ T_{21}(t) & T_{22}(t) & \cdots & T_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ T_{n1}(t) & T_{n2}(t) & \cdots & T_{nn}(t) \end{pmatrix} \begin{pmatrix} p_1(t) \\ p_2(t) \\ \vdots \\ p_n(t) \end{pmatrix}$$

$$\Longrightarrow P(t+dt) = T(t)P(t)$$

On discrétise le temps : $t = n \in \mathbb{N}$
$\forall n \in \mathbb{N}$,

$$P_{n+1} = T_n P_n$$

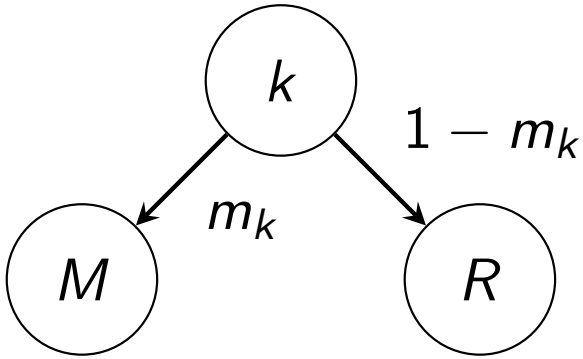
Et on a alors $\forall n \in \mathbb{N}^*$,

$$P_n = (\prod_{k=0}^{n-1} T_k) P_0$$

Où les individus initialement infectés fournissent P_0.

Soient :

- τ_k La durée caractéristique d'infection de l'individu k.
- m_k La probabilité de décès de l'individu k à un instant donné de l'infection, durant τ_k.



Soit M_k la probabilité que k soit décédé durant τ_k de la maladie.

$$1 - M_k = \lim_{\delta t \to 0} (1 - m_k \delta t)^{\tau_k / \delta t} = e^{-m_k \tau_k}$$

$$\Longrightarrow \quad M_k = 1 - e^{m_k \tau_k}$$

$\forall k \in \Omega$,

$$q_k(t) = \begin{cases} p_k(t - \tau_k)M_k & \text{si } t > \tau_k \\ 0 & \text{sinon} \end{cases}$$

$$r_k(t) = \begin{cases} p_k(t - \tau_k)(1 - M_k) & \text{si } t > \tau_k \\ 0 & \text{sinon} \end{cases}$$

D'où,

$$I_k(t) \sim B(p_k(t)), M_k(t) \sim B(q_k(t)), R_k(t) \sim B(r_k(t))$$

Soit $k \in \{1, ..n\}$,
On note $(C_n^k(p))_{1 \leq p \leq C_n^k}$ la suite des combinaisons de k éléments parmi n.

$$\mathbb{P}(I(t) = k) = \sum_{p=1}^{C_n^k} \prod_{i \in C_n^k(p)} p_i(t) \prod_{i \in \Omega / C_n^k(p)} (1 - p_i(t))$$

$$\mathbb{E}(I(t)) = \sum_{k=1}^{n} k\mathbb{P}(I(t) = k)$$

$$\mathbb{E}(I(t)) = \sum_{k=1}^{n} \sum_{p=1}^{C_n^k} k \prod_{i \in C_n^k(p)} p_i(t) \prod_{i \in \Omega / C_n^k(p)} (1 - p_i(t))$$

$$\mathbb{E}(M(t)) = \sum_{k=1}^{n} \sum_{p=1}^{C_n^k} k \prod_{i \in C_n^k(p)} q_i(t) \prod_{i \in \Omega / C_n^k(p)} (1 - q_i(t))$$

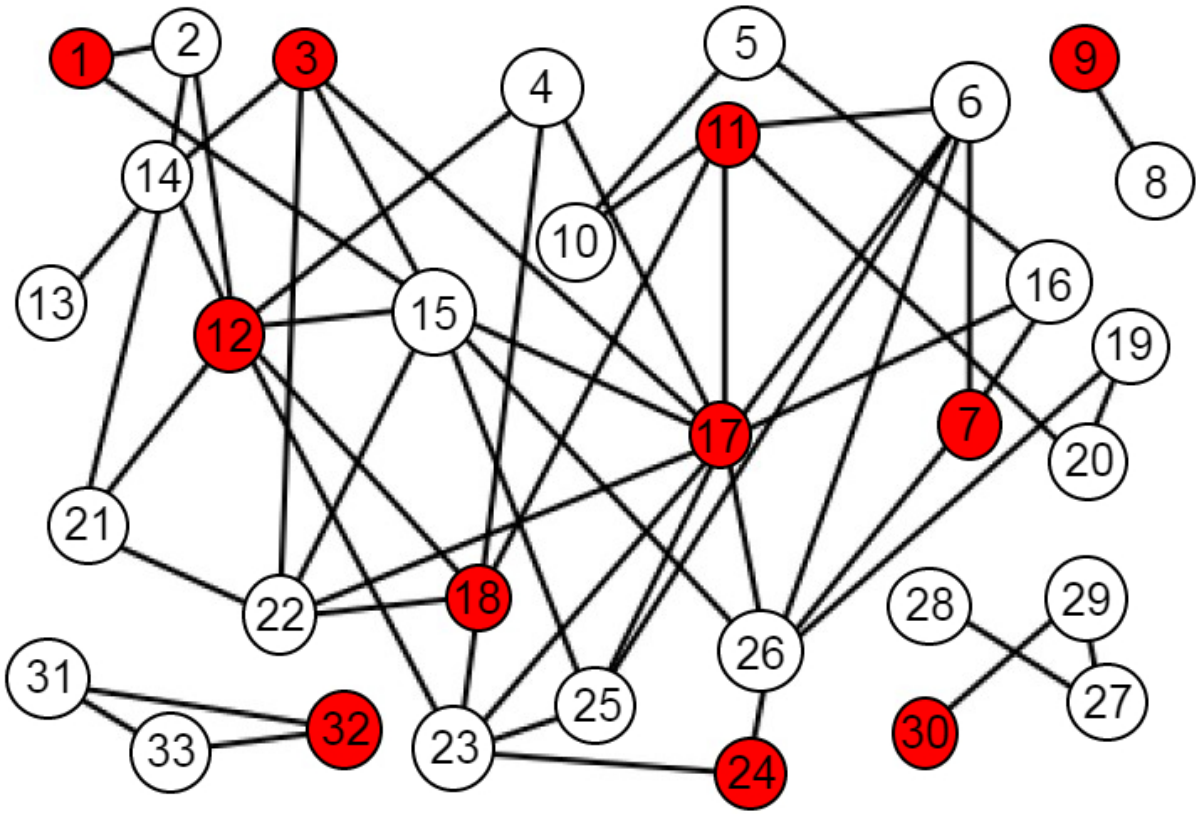
$$\mathbb{E}(R(t)) = \sum_{k=1}^{n} \sum_{p=1}^{C_n^k} k \prod_{i \in C_n^k(p)} r_i(t) \prod_{i \in \Omega / C_n^k(p)} (1 - r_i(t))$$

En considérant $P(t)$ la variable aléatoire du nombre des porteurs, on a :

$$\mathbb{E}(P(t)) = \mathbb{E}(I(t)) - \mathbb{E}(R(t)) - \mathbb{E}(M(t))$$

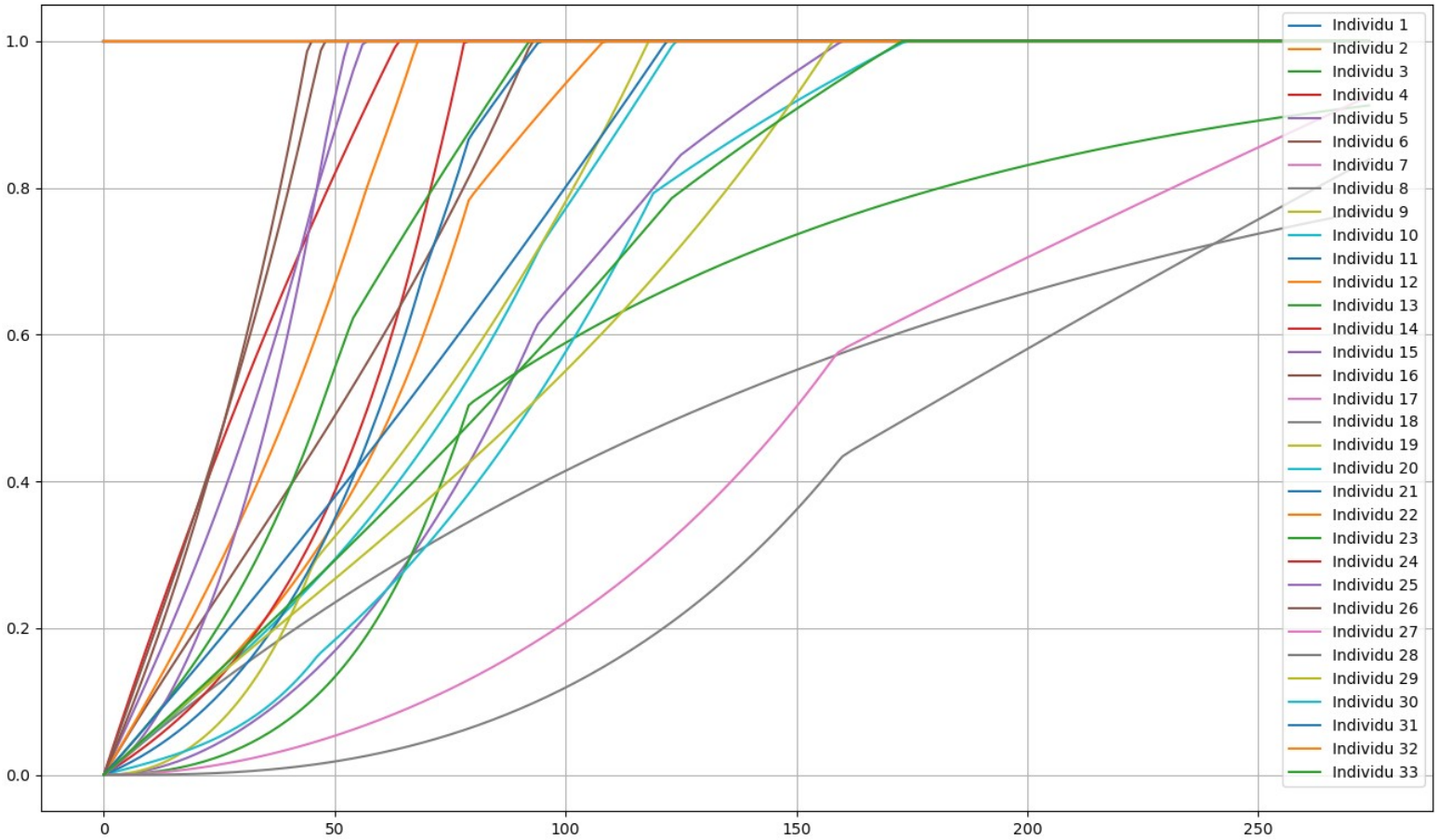
Simulation numérique des résultats

Graphe de la population considérée



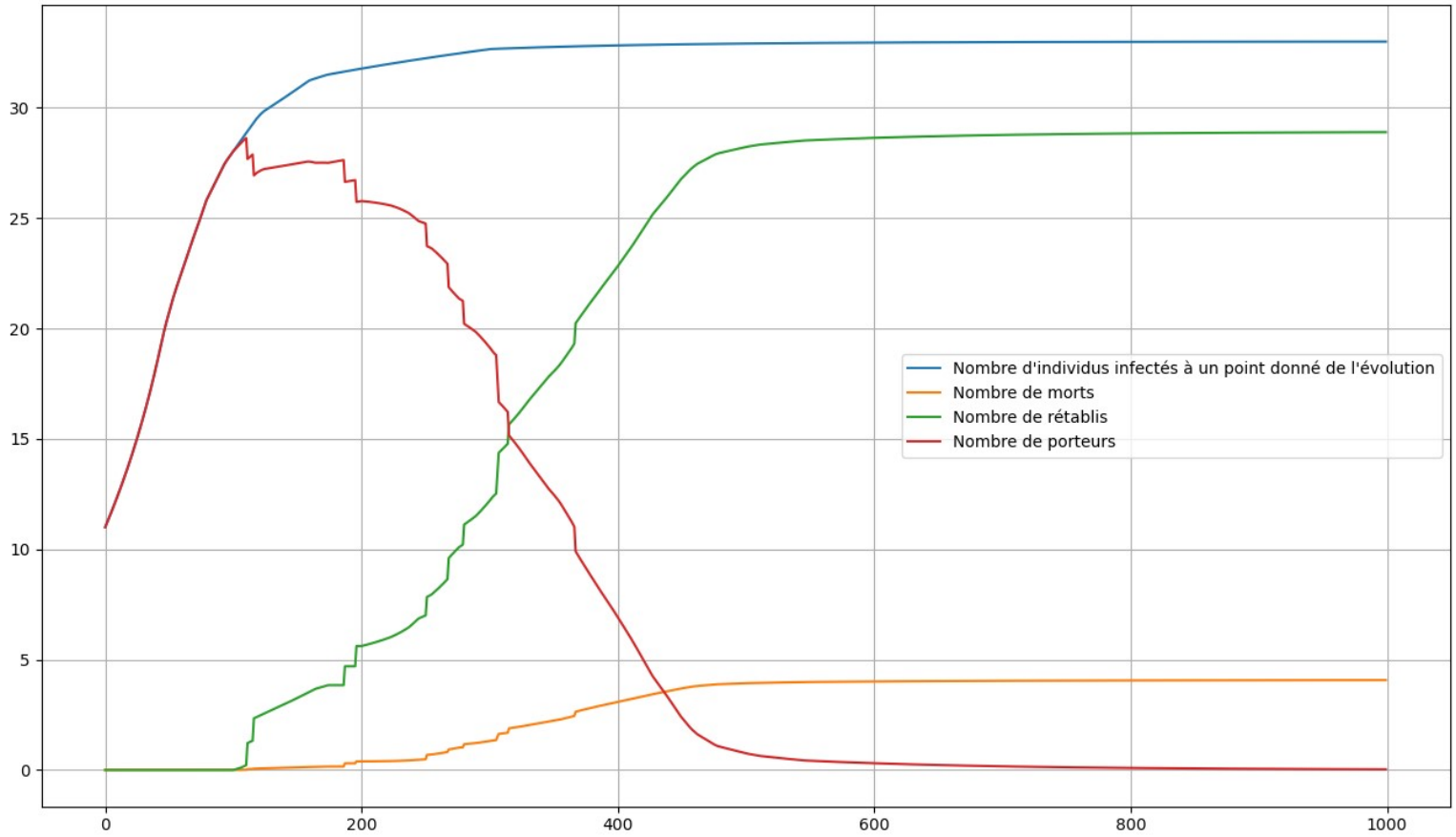
Simulation numérique des résultats

Evolution des probabilités d'infection



Simulation numérique des résultats

Evolution du nombre de porteurs, d'infectés, de décédés et de rétablis



Contrôle de la propagation de l'épidémie

Définition : Nombre de reproduction de base R_0

En épidémiologie, le nombre de reproduction de base d'une maladie est défini comme le nombre moyen de cas attendus directement générés par un cas dans une population sensible à l'infection.

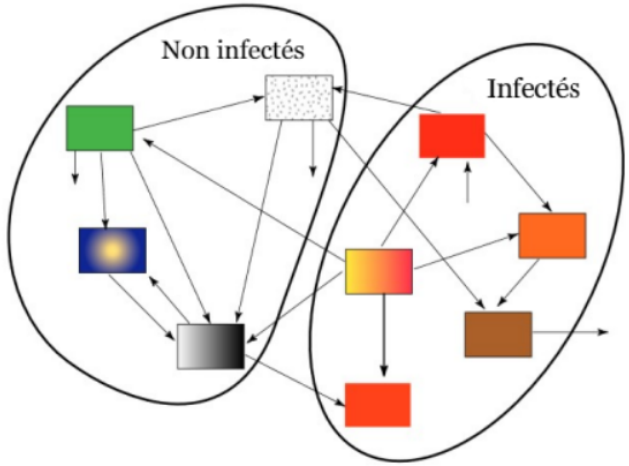
Théorème de Stabilité d'un système épidémiologique

Si $R_0 < 1$, alors la maladie disparaît de la population au bout d'un certain temps.
Si $R_0 \geq 1$, alors la maladie reste endémique dans la population.

Modèle théorique de contrôle d'épidémie

Détermination de R_0 : Approche de la matrice de prochaine génération

Soit $n \in \mathbb{N}^*$, on définit n compartiments distincts pour les individus de la population,



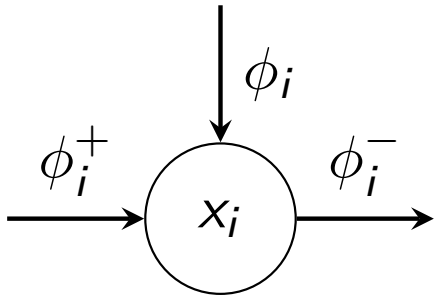
$$x = (\underbrace{x_1, \ldots, x_p}_{\text{non infectés}}, \underbrace{x_{p+1}, \ldots, x_n}_{\text{infectés}}) \in \mathbb{R}^n$$

Avec $x_i \geq 0$ correspond au nombre d'individus dans chaque compartiment.

On décrit la propagation de la maladie par $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}^n$.

$$\dot{x}(t) = f(x(t))$$

En décomposant les flux :



On pourra alors décomposer f de la forme suivante :

$$f = \phi + \phi^+ + \phi^-$$

Théorème de La matrice de prochaine génération (Fondamental)

Soit un point d'équilibre sans maladie, $x^* = (x_1^*, ..., x_p^*, 0, ..., 0)$,
Soient les matrices,

$$\mathbb{F} = \left(\frac{\partial \phi_i}{\partial x_j}(x^*)\right)_{p+1 \leq i,j \leq n}, \mathbb{V} = \left(\frac{\partial \phi_i^+ - \phi_i^-}{\partial x_j}(x^*)\right)_{p+1 \leq i,j \leq n}$$

Le nombre de reproduction de base lié au point d'équilibre x^* est :

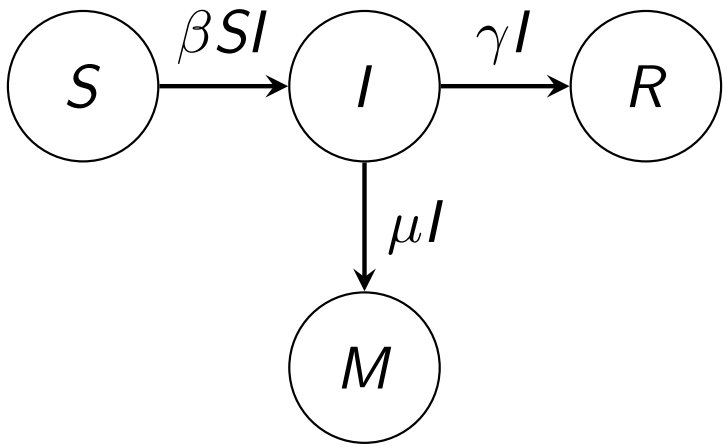
$$R_0 = \rho(-\mathbb{F}\mathbb{V}^{-1})$$

Avec : ρ désigne le rayon spectral.

Application à notre modélisation

Modélisation choisi (Modèle SRMI)

On dispose de 4 compatiments $x = (S, R, M, I)$ pour la population:



Avec :

- μ le taux de mortalité de la maladie.
- β le taux d'infection par contact.
- γ le taux de guérison de la maladie.

Application à notre modélisation

Paramètres de contrôle

Soient u_1, u_2, u_3 les paramètres de contrôle de la propagation définis dans [0,1] comme suit :

- u_1 : Degré de confinement et mesures de sécurité (Réduction des contacts).
- u_2 : Rétablissement (Prise en charge des infectés et traitement).
- u_3 : Prévention des Décès (Intervention dans les cas critiques).

En intégrant ces paramètres, on obtient le système d'ED suivant :

$$\begin{cases} \dot{S} = -(1-u_1)\beta SI \\ \dot{I} = (1-u_1)\beta SI - (1-u_3)\mu I - (1+u_2)\gamma I \\ \dot{R} = (1+u_2)\gamma I \\ \dot{M} = (1-u_3)\mu I \end{cases}$$

Application à notre modélisation

Calcul de R_0

En appliquant le théorème de la matrice de prochaine génération :

Pour un état d'équilibre sans malades $x^* = (S^*, R^*, M^*, 0)$, on retrouve pour le modèle considéré :

$$R_0 = \frac{(1-u_1)\beta}{(1+u_2)\gamma+(1-u_3)\mu} S^*$$

Notre but dorénavant est d'ajuster les paramètres de contrôle de sorte à avoir $R_0 < 1$ pour un équilibre où l'on a S^* sains.

Hypothèses :

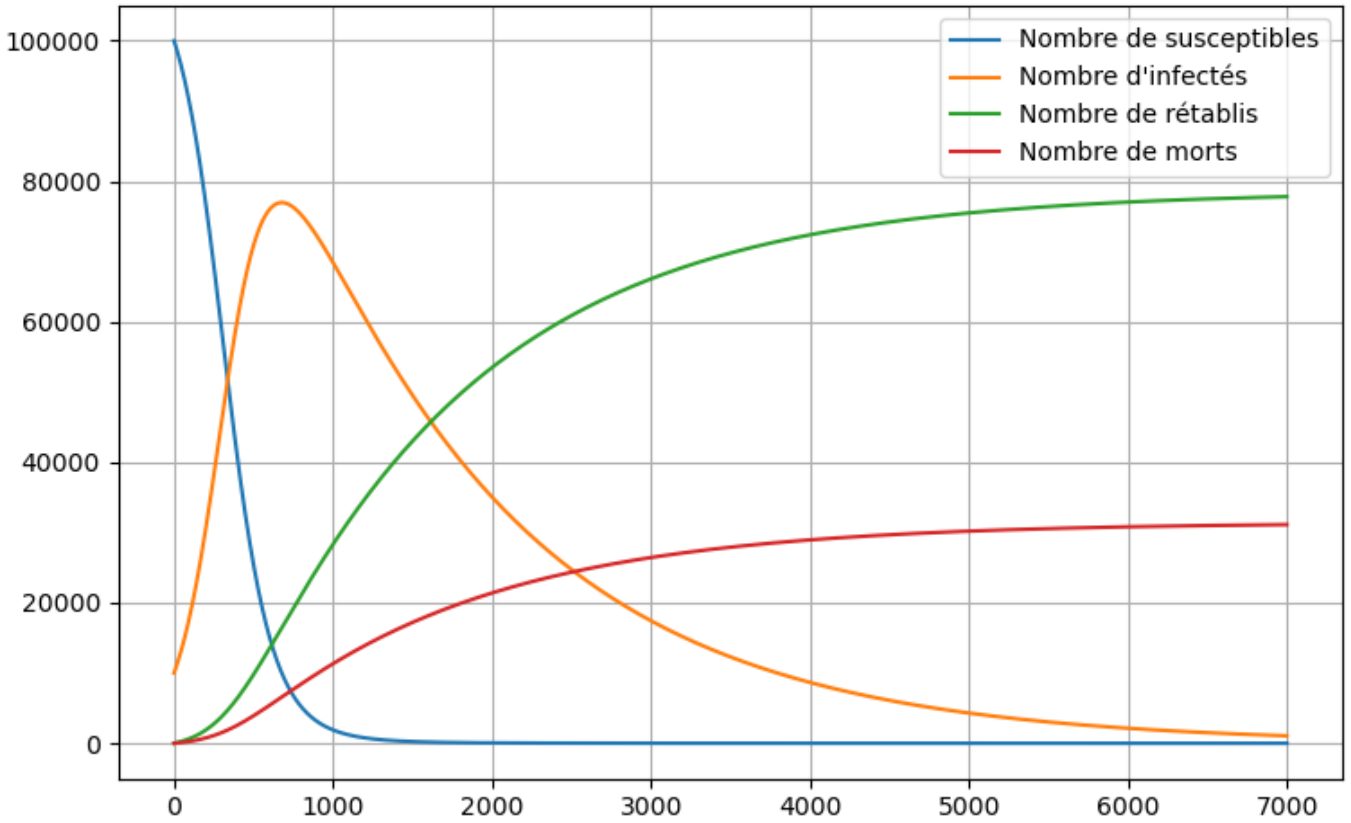
- On se donne une population de 110.000 individus dont 10.000 sont infectés.
- Dans un premier temps, nous simulons la maladie avec des paramètres de contrôle nuls, puis nous simulons leur effet sur la propagation de la maladie.

Simulation numérique des résultats

Evolution de la maladie au sein de la population

Cas 1 : $R_0 = 10$

$u_1 = u_2 = u_3 = 0$ et $\beta = 7.10^{-8}, \gamma = 5.10^{-4}, \mu = 2.10^{-4}$

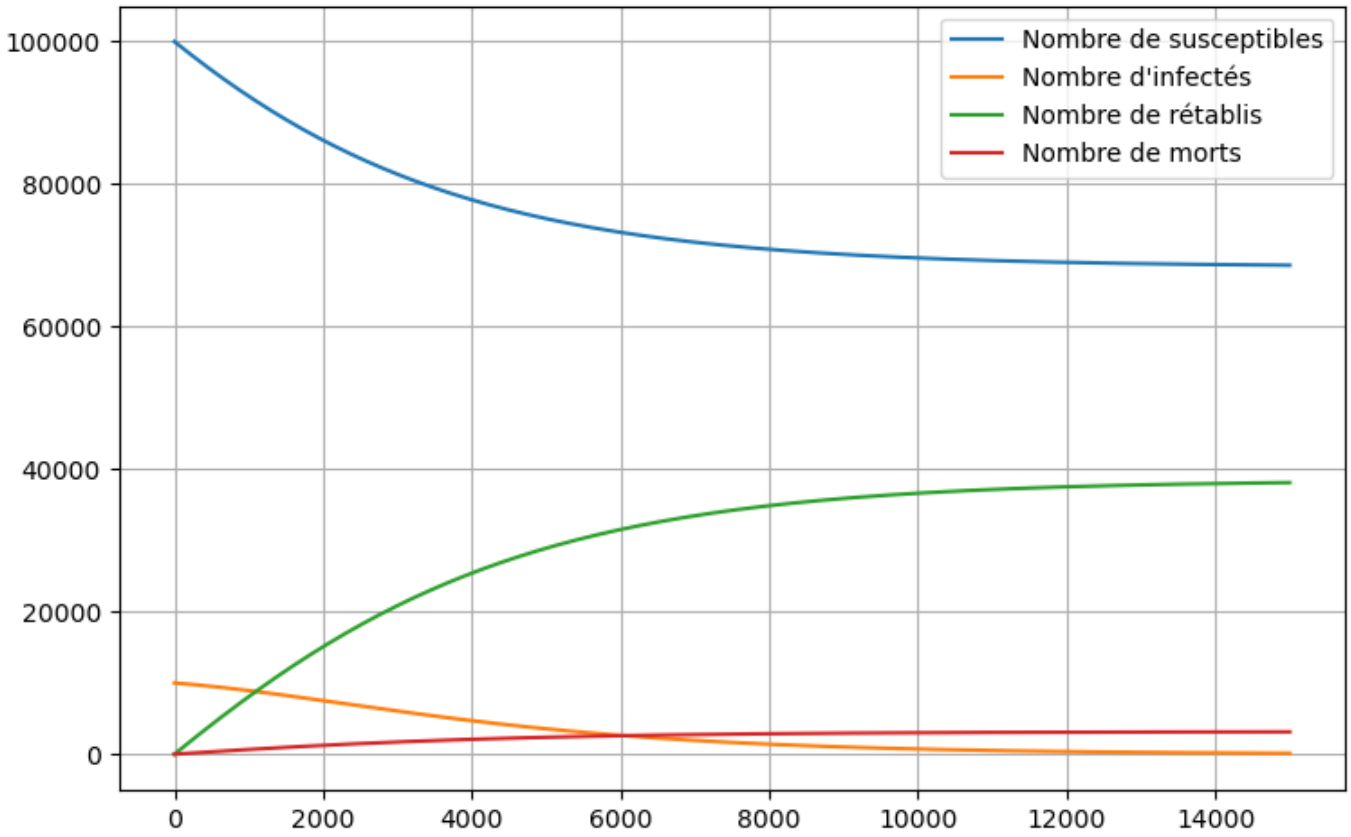


Simulation numérique des résultats

Evolution de la maladie au sein de la population

Cas 2 : $R_0 = 0.91$

$u_1 = 0.88$, $u_2 = 0.7$, $u_3 = 0.65$ et $\beta = 7.10^{-8}, \gamma = 5.10^{-4}, \mu = 2.10^{-4}$



Conclusion

Développement du calcul de R_0

D'après le système différentiel, pour $x = (S, R, M, I)$:

$$\phi(x) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ (1-u_1)\beta SI \end{pmatrix}, \phi^+(x) = \begin{pmatrix} 0 \\ (1+u_2)\gamma I \\ (1-u_3)\mu I \\ 0 \end{pmatrix}, \phi^-(x) = \begin{pmatrix} (1-u_1)\beta SI \\ 0 \\ 0 \\ (1+u_2)\gamma I + (1-u_3)\mu I \end{pmatrix}$$

Pour un point d'équilibre sans maladies $x^* = (S^*, R^*, M^*, 0)$
En appliquant le théorème :

$$\mathbb{F} = (1-u_1)\beta S^* \text{ et } \mathbb{V} = -((1+u_2)\gamma + (1-u_3)\mu)$$

Ainsi :

$$R_0 = \frac{(1-u_1)\beta}{(1+u_2)\gamma+(1-u_3)\mu} S^*$$

Commentaires :

— Il en découle ainsi que pour un équilibre avec plus d'individus sains, il est nécessaire d'adopter des paramètres de contrôle plus grands pour assurer cet équilibre.

1 Théorème 1 : Matrices de Metzler

1.1 Définitions :

Soit $A = (a_{i,j})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}} \in M_{n,m}(\mathbb{R})$,

- A est une matrice non négative si $\forall i \in \{1, .., n\}$, $\forall j \in \{1, .., m\}$ on a $a_{i,j} \geq 0$, c'est à dire que toute ses entrées non négatives. Nous notons une telle matrice $A \geq 0$.
- A est une matrice positive si A est non-négative et $\exists i \in \{1, .., n\}$, $\exists j \in \{1, .., m\}$ tel que $a_{k,l} > 0$, c'est à dire ayant au moins un coefficient strictement positif.
- A est une matrice strictement positive si $\forall i \in \{1, .., n\}$, $\forall j \in \{1, .., m\}$ on a $a_{i,j} > 0$, c'est à dire tous ses coefficients sont strictement positifs. Nous noterons une telle matrice $A >> 0$.
- On dit que A est une matrice de Metzler si $\forall (i, j) \in \{1, ..., n\}^2$ tels que $i \neq j$ on a $a_{i,j} \geq 0$.
- On dit qu'une matrice M est asymptotiquement stable si $\rho(M) < 1$ avec ρ le rayon spectral.

1.2 Théorème de Metzler :

Si A est une matrice de Metzler, les conditions suivantes sont équivalentes :

- La matrice de Metzler A est asymptotiquement stable.
- La matrice de Metzler A est inversible et $-A^{-1} \geq 0$.
- Si b est un vecteur tel que $b >> 0$ alors il existe $x >> 0$ tel que $Ax + b = 0$.
- Il existe $c > 0$ tel que $Ac << 0$.
- Il existe $c >> 0$ tel que $Ac << 0$.

2 Théorème 2 : Théorème de Perron-Frobenius

Le rayon spectral ρ d'une matrice positive $A \geq 0$ est une valeur propre à laquelle est associé un unique vecteur x de norme 1 à coordonnées strictement positives tel que $Ax = \rho x$.

Ce vecteur est unique, à un coefficient multiplicatif près car ρ est une valeur propre simple c'est à dire d'espace propre de dimension 1.

3 Théorème 3 : Critère de Stabilité de Lyapunov

3.1 Définitions :

Soit $\Omega \subset \mathbb{R}^n$ ouvert, connexe et $f : \Omega \to \mathbb{R}^n$ fonction de classe C^1 et $x : t \in \mathbb{R}^+ \to \mathbb{R}^n$
Soit un système différentiel autonome défini par $\dot{x}(t) = f(x(t))$.

— Nous appelons x^* un point d'équilibre ou fixe si $f(x^*) = 0$.

— Soit t_0 l'origine des temps, on dit que x^* est stable si,

$$\forall \epsilon > 0, \exists \delta > 0, \forall t \geq t_0, \ \|x(t_0) - x^*\| < \delta \implies \|x(t) - x^*\| < \epsilon$$

— On dit que x^* est asymptotiquement stable si,

$$x^* \text{ est stable et } \exists \delta > 0, \ \|x(t_0) - x^*\| < \delta \implies x(t) \xrightarrow{t \to +\infty} x^*$$

Remarquer que les points stables ne sont pas forcément asymptotiquement stables.

3.2 Théorème de Lyapunov :

Soit x^* un point d'équilibre ie $f(x^*) = 0$. Si $f \in C^1(\Omega, \mathbb{R}^n)$ tel que $Df(x^*) \in M_n(\mathbb{R})$ a toutes ses valeurs propres complexes de partie réelle strictement négative, alors x^* est un point d'équilibre attractif.

C'est à dire que, $\exists\ r > 0$ tel que pour $x_0 \in \mathbb{R}^n$ proche de x^* tel que $\|x_0 - x^*\| \leq r$, la solution de :

$$\begin{cases} u'(t) = f(u(t)) \\ u(t_0) = x_0 \end{cases}$$

vérifie $u(t) \xrightarrow{t \to +\infty} x^*$, x^* est donc asymptotiquement stable. De plus, cette convergence se fait à une vitesse exponnentielle.

4 Théorème 4 : Théorème de Varga

Soit une matrice de Metzler A. Alors pour toute décomposition de A de la forme $A = F + V$, où $F \geq 0$ et V une matrice de Metzler asymptotiquement stable, les deux propositions suivantes sont équivalentes :

- A est asymptotiquement stable.
- $\rho(-FV^{-1}) < 1$.

Démonstration du théorème de Stabilité d'un système épidémiologique

Notons pour $A \in M_n(\mathbb{R})$,

$$\alpha(A) = max\{Re(\lambda), \forall \lambda \in Sp_{\mathbb{C}}(A)\}$$

D'après le théorème 1 de Metzler, $-\mathbb{F}\mathbb{V}^{-1}$ est positive, alors par le théorème 2 de Perron-Frobenius, $R_0 = \rho(-\mathbb{F}\mathbb{V}^{-1})$ est une valeur propre de $-\mathbb{F}\mathbb{V}^{-1}$. En utilisant le théorème 3 de Lyapunov, alors soit un point d'équilibre sans maladies x^*, c'est un point asymptotiquement stable si $\alpha(Df(x^*)) < 0$ et asymptotiquement instable si $Df(x^*) \geq 0$. Sachant $f = \phi + \nu$, $\alpha(\mathbb{V}) < 0$ et $\alpha(J_1) < 0$

Alors par le théorème 4 de Varga :

$$\alpha(\mathbb{F} + \mathbb{V}) < 0 \iff R_0 < 1 \implies \alpha(Df(x^*)) < 0 \iff R_0 < 1$$

Ainsi la stabilité équivaut $R_0 < 1$.

Démonstration du théorème de la matrice de prochaine génération

- Sachant que R_0 est le nombre d'infections secondaires produites par une seule infection typique dans une population sensible, alors comment définir R_0 lorsqu'il existe plusieurs types d'individus infectés. Le concept clé est que nous devons maintenant faire la moyenne du nombre prévu de nouvelles infections sur tous les types d'infections possibles.
- Supposons que nous ayons un système dans lequel plusieurs individus sont infectés, la matrice de la prochaine génération est définie comme la matrice carrée K dans laquelle l'élément $k_{i,j}$, est le nombre attendu d'infections secondaires de type i causées par un seul individu infecté par le type j.
- Ayant la matrice K, il est naturel de penser que R_0 représentera la plus grande entrée de K. le nombre de base de la reproduction est donné par le rayon spectral de K, la valeur propre dominante de K.

Soit x^* un point d'équilibre sans maladies, en posant $\nu = \phi^+ - \phi^-$,

$$Df(x^*) = D\phi(x^*) + D\nu(x^*)$$

$$D\phi(x^*) = \begin{pmatrix} 0 & 0 \\ 0 & \mathbb{F} \end{pmatrix}, \; D\nu(x^*) = \begin{pmatrix} J_1 & J_2 \\ 0 & \mathbb{V} \end{pmatrix}$$

Avec :

$$\mathbb{F} = \left(\frac{\partial \phi_i}{\partial x_j}(x^*)\right)_{p+1\leq i,j\leq n}, \mathbb{V} = \left(\frac{\partial \nu_i}{\partial x_j}(x^*)\right)_{p+1\leq i,j\leq n}, J_1 = \left(\frac{\partial \nu_i}{\partial x_j}(x^*)\right)_{1\leq i,j\leq p}, J_2 = \left(\frac{\partial \nu_i}{\partial x_j}(x^*)\right)_{\substack{1\leq i\leq p \\ p+1\leq j\leq n}}$$

Remarquons que $\mathbb{F} \geq 0$ et comme x^* point d'équilibre , $\mathbb{V}$ est une matrice de Metzler.

Approximation au voisinnage de l'équilibre :

De plus $\phi = 0$, car x^* est sans maladies, nous sommes près de l'équilibre le comportement du système est approximé par le système linéarisé, on se place ainsi dans le cadre de cette approximation.

$$\dot{x} = D\nu(x^*)(x - x^*)$$

$$(\dot{x}_1, ..., \dot{x}_n)^T = \begin{pmatrix} J_1 & J_2 \\ 0 & \mathbb{V} \end{pmatrix} (x_1 - x_1^*, ..., x_m - x_m^*, x_{m+1}, ..., x_n)^T$$

En posant $x_{infectés}(t) = (x_{m+1}, ..., x_n)^T$,

$$\dot{x}_{infectés}(t) = \mathbb{V}\ x_{infectés}(t)$$

$$x_{infectés}(t) = exp(\mathbb{V}t)\ x_{infectés}(0)$$

Les individus infectés de l'état initial jusqu'à l'équilibre est :

$$\int_0^{+\infty} x_{infectés}(t)\,\mathrm{d}t = \int_0^{+\infty} exp(\mathbb{V}t)\,\mathrm{d}t\ x_{infectés}(0) = -\mathbb{V}^{-1}x_{infectés}(0)$$

Ces individus cont générer d'autres cas selon la matrice d'infection $\mathbb{F}$, le nombre total d'infections produites est :

$$\int_0^{+\infty} \mathbb{F}\ x_{infectés}(t)\,\mathrm{d}t = -\mathbb{F}\mathbb{V}^{-1}x_{infectés}(0)$$

D'où :

$$K = -\mathbb{F}\mathbb{V}^{-1} \implies R_0 = \rho(-\mathbb{F}\mathbb{V}^{-1})$$

Annexe

Code Python : $1^{ère}$ Simulation

```python
import numpy as np
from matplotlib import pyplot as plt
import random

##Simulation des résultats théoriques##

#On se propose d'implémenter la structure de graphes à l'aide d'une liste d'adjacence#

#Ainsi un graphe G=[[s_1,[voisin_1,....,voisin_n],[r_11,....,r_1n],tau_1,m_1],.....,[s_n,[voisin_1,.......,voisin_m],
[rn1,....,r_nm],tau_n,m_n]]#

#s_i re présente un individu, la liste [voisin_1,....,voisin_n] ses voisins(individus avec lesquels il est en contact),
[r_i1,....,r_in] la probabilité d'infection par les autres individus, tau_i sa durée caractéristique d'infection et m_i la
probabilité de décès à un instant donné de l'infection#

def graphe_vide(n:int)->list:
    '''Cette fonction permet de créer un graphe vide avec n noeuds et 0 liens entre eux'''
    res=[]
    for i in range(n):
        res+=[[i+1,[],[]]]
    return res

def voisins(G:list,i:int)->list:
    '''Cette fonction renvoie la liste des voisins d'un noeud i dans un graphe G'''
    res=G[i-1][1]
    return res

def creer_lien(G,i,j):
    '''Cette procédure permet de créer un lien entre i et j dans un graphe G'''
    G[i-1][1]+=[j]
    G[j-1][1]+=[i]
    G[i-1][1].sort()
    G[j-1][1].sort()

def random_matrix(n,inf,sup):
    '''Cette fonction renvoie une matrice avec des coefficients aléatoirement choisis dans une plage bien définie'''
    res=[]
    for i in range(n):
        line=[]
        for j in range(n):
            if i==j:
                line+=[0]
            else:
                line+=[random.uniform(inf,sup)]
        res+=[line]
    return res
```

Annexe

Code Python : $1^{ère}$ Simulation

```python
def random_column(n,type,inf,sup):
    '''Cette fonction renvoie une matrice colonne avec des coefficients aléatoirement choisis selon le type demandé dans une plage bien définie'''
    res=[]
    if type=='reel':
        for i in range(n):
            res+=[random.uniform(inf,sup)]
    if type=='entier'
        for i in range(n):
            res+=[random.randint(inf,sup)]
    return res

def show_matrix(T):
    '''Cette fonction permet de visualiser une matrice'''
    for line in T:
        print(line)

def random_coeff_transmission(G,inf,sup,inf1,sup1,inf2,sup2):
    '''Cette fonction permet d'affecter au graphe G des probabilités aléatoires de transmission dans un intervalle dt entre les individus du graphe, les durées caractéristiques de l'infection et les porbabilités de décès, ces coefficients sont choisies dans un intervalle donné'''
    R=random_matrix(len(G),inf,sup)
    T=random_column(len(G),'entier',inf1,sup1)
    M=random_column(len(G),'reel',inf2,sup2)
    for i in range(len(G)):
        vois=voisins(G,G[i][0])
        for j in vois:
            G[i][2]+=[R[i][j-1]]
        G[i]+=[T[i],M[i]]
    return G

def transmission_probability(G,t,i,j):
    '''Cette fonction renvoie la probabilité de transmission entre les individus i et j d'un graphe G à l'instant t'''
    vois=voisins(G,i)
    if j not in vois:
        return 0
    else:
        return 1-np.exp(-(G[i-1][2][G[i-1][1].index(j)])*t)
```

Annexe

Code Python : $1^{ère}$ Simulation

```python
def transmission_matrix(G,t):
    '''Cette fonction renvoie la matrice de transmission du graphe G à l'instant t'''
    res=[]
    for i in range(len(G)):
        ligne=[]
        for j in range(len(G)):
            if j+1 not in G[i][1]:
                ligne+=[0]
            else:
                ligne+=[transmission_probability(G,t,i+1,j+1)]
        res+=[ligne]
    return res

def matrix_x_column(A,x):
    '''Cette fonction renvoie la matrice colonne résultante du produit d'une matrice carrée et d'une matrice colonne'''
    n=len(A)
    res=[]
    for i in range(n):
        sum=0
        for j in range(n):
            sum+=A[i][j]*x[j]
        res+=[sum]
    return res

def infection_probabilities(G,t,p0):
    '''Cette fonction renvoie une liste contenant les listes des probabilités d'infection des individus de l'instant initial à l'instant t pour la population G à partir d'un état initial p0'''
    res=[p0]
    infected=[]
    for j in range(len(res[0])):
        if res[len(res)-1][j]==1:
            infected+=[j]
    i=1
    while i<t:
        T=transmission_matrix(G,i)
        a=matrix_x_column(T,res[len(res)-1])
        for j in range(len(a)):
            if a[j]>1 or j in infected:
                a[j]=1
        res+=[a]
        i+=1
    return res
```

Annexe

Graphe de la population considérée : $1^{ère}$ Simulation

```
[1, [2, 15], []]
[2, [1, 12, 14], []]
[3, [14, 15, 17, 22], []]
[4, [12, 17, 18], []]
[5, [10, 16], []]
[6, [7, 11, 17, 25, 26], []]
[7, [6, 16, 26], []]
[8, [9], []]
[9, [8], []]
[10, [5, 11], []]
[11, [6, 10, 17, 18, 20], []]
[12, [2, 4, 14, 15, 18, 21, 23], []]
[13, [14], []]
[14, [2, 3, 12, 13, 21], []]
[15, [1, 3, 12, 17, 22, 25, 26], []]
[16, [5, 7, 17], []]
[17, [3, 4, 6, 11, 15, 16, 22, 23, 25, 26], []]
[18, [4, 11, 12, 22, 23], []]
[19, [20, 26], []]
[20, [11, 19], []]
[21, [12, 14, 22], []]
[22, [3, 15, 17, 18, 21], []]
[23, [12, 17, 18, 24, 25], []]
[24, [23, 26], []]
[25, [6, 15, 17, 23], []]
[26, [6, 7, 15, 17, 19, 24], []]
[27, [28, 29], []]
[28, [27], []]
[29, [27, 30], []]
[30, [29], []]
[31, [32, 33], []]
[32, [31, 33], []]
[33, [31, 32], []]
```

Annexe

Matrice aléatoire de la population considérée : $1^{ère}$ Simulation


```
[1, [2, 15], [0.0032763815471536883, 0.0042655887093318925], 116, 0.00011968838200846785]
[2, [1, 12, 14], [0.0039769404843512 48, 0.0010960674036354506, 0.007140472666827515], 370, 0.00019995927430859183]
[3, [14, 15, 17, 22], [0.002207223683235355, 0.009066156079414744, 0.0036916781007778157, 0.0027443553376940498], 111, 0.00023120541873833405]
[4, [12, 17, 18], [0.008257652771116626, 0.004682273756500029, 0.0063555735342235 07], 100, 0.0004699928259739991]
[5, [10, 16], [0.0027644794193266327, 0.0064406654617619715], 243, 0.0009404668277020307]
[6, [7, 11, 17, 25, 26], [0.008204939000534792, 0.0015914259950801659, 0.007949924604537365, 0.006537521271534095, 0.0016223237711327555], 255, 0.0006190895377769641]
[7, [6, 16, 26], [0.004429055875680989, 0.004622692072341597, 0.008440875070694652], 268, 0.0004089873416542379]
[8, [9], [0.0053511213866457 82], 355, 0.0007955673867904787]
[9, [8], [0.0049305492938734 29], 306, 0.0005037619343484045]
[10, [5, 11], [0.007541087310819459, 0.005559738796635799], 253, 0.0003919835869047986]
[11, [6, 10, 17, 18, 20], [0.009476814181450768, 0.0011756884166070033, 0.009330636044797357, 0.006595582995271426, 0.0069883525515195945], 307, 0.00041108511254529985]
[12, [2, 4, 14, 15, 18, 21, 23], [0.003559439823935663, 0.002377808198991901, 0.001235800570399022, 0.00800452773280254, 0.007514548548132448, 0.0017019101974703173, 0.0056
83467446378362], 251, 0.0008576317215331715]
[13, [14], [0.00888816501892252], 288, 0.00010387278187533277]
[14, [2, 3, 12, 13, 21], [0.005115171663798195, 0.002353386810406758, 0.0016363526617590985, 0.0038375034705428516, 0.006958903459154055], 349, 0.00027240977626992154]
[15, [1, 3, 12, 17, 22, 25, 26], [0.0013981874392566783, 0.0013014573612684633, 0.0013652123107169693, 0.009318682164134204, 0.002351446952736666, 0.002247300589376015, 0.0
030523645008625473], 289, 0.0002564804554816647]
[16, [5, 7, 17], [0.007699213318544966, 0.008858830514271141, 0.0016571733526110526], 237, 0.00017277408813322226]
[17, [3, 4, 6, 11, 15, 16, 22, 23, 25, 26], [0.004377441678360239, 0.006209825262222289, 0.005549086630562283, 0.004690538982115096, 0.009436511555775781, 0.004350300590543
201, 0.0036977505486024105, 0.004350981214523371, 0.008215801001002485, 0.0035149793397061223], 315, 0.0006673495268805579]
[18, [4, 11, 12, 22, 23], [0.009250948445457246, 0.009459857936625123, 0.004436341282218397, 0.007490029895414177, 0.008832932225242868], 367, 0.0005167982699429601]
[19, [20, 26], [0.008519243457478003, 0.006098380001968974], 344, 0.0003422471062 4065777]
[20, [11, 19], [0.0016396759277304087, 0.008032622719388111], 328, 0.000556757378561663]
[21, [12, 14, 22], [0.0026301647246975536, 0.006314957874743175, 0.004272979310412487], 291, 0.0008618923918268285]
[22, [3, 15, 17, 18, 21], [0.0018607234548556999, 0.002928435796325076, 0.007569208141819231, 0.0011571624513687484, 0.006388335603277425], 106, 0.0007031422313059147]
[23, [12, 17, 18, 24, 25], [0.0020736653877131 58, 0.0013684241577810 74, 0.0019272742277894773, 0.0016569289127067208, 0.005605561114374 5725], 364, 0.0003846468961844034]
[24, [23, 26], [0.009911363689031592, 0.0017923115817388728], 187, 0.0008494163936740655]
[25, [6, 15, 17, 23], [0.006142256701466365, 0.008618298416245031, 0.00430167972436005, 0.007284450874223072], 223, 0.0008927477020606137]
[26, [6, 7, 15, 17, 19, 24], [0.006291246730536206, 0.0059665623440478014, 0.005498340300654059, 0.002291392432607046, 0.004791207861975831, 0.006327756482415834], 200, 0.0
0160411716114037 46]
[27, [28, 29], [0.0011167044876294243, 0.004454062953590836], 245, 0.0007902821432313956]
[28, [27], [0.008767874017005628], 175, 0.0006265053238213759]
[29, [27, 30], [0.008167437077362952, 0.005744931608637256], 300, 0.0008563575613351633]
[30, [29], [0.0030574460350339464], 280, 0.00050927073237675]
[31, [32, 33], [0.007135561973617852 5, 0.006435384151283527], 354, 0.0007599170910484455]
[32, [31, 33], [0.004637376767915197, 0.008596283448421317], 196, 0.0004494241538436099]
[33, [31, 32], [0.0027580807972997544, 0.0056090638789691345], 338, 0.0001804918927120 4683]
```

Annexe

Code python des Courbes : 1ère Simulation

```
##Evolution du nombre d'infectés et de susceptibles##
Nombre_infected=[]
for instant in liste_proba_infection:
    S=0
    for j in instant:
        S+=j
        S1=33-S
    Nombre_infected+=[S]
    Nombre_susc+=[S1]

plt.plot(t,Nombre_infected,label="Nombre d'infectés")
plt.plot(t,Nombre_susc,label="Nombre de susceptibles")
plt.legend()
plt.grid()
plt.show()
##Evolution du nombre de morts et de rétablis##
Nombre_morts=[]
Nombre_rétablis=[]
for instant in range(len(evolution_morts[0])):
    S1=0
    for individu in evolution_morts:
        S1+=individu[instant]
    Nombre_morts+=[S1]
for instant in range(len(evolution_retab[0])):
    S2=0
    for individu in evolution_retab:
        S2+=individu[instant]
    Nombre_rétablis+=[S2]

plt.plot(t,Nombre_morts,label="Nombre de morts")
plt.plot(t,Nombre_rétablis,label="Nombre de rétablis")
plt.legend()
plt.grid()
plt.show()
##Evolution du nombre de porteurs##
Nombre_porteurs=[]
for i in range(len(Nombre_infected)):
    res=Nombre_infected[i]-Nombre_morts[i]-Nombre_rétablis[i]
    Nombre_porteurs+=[res]

plt.plot(t,Nombre_porteurs,label="Nombre de porteurs")
plt.legend()
plt.grid()
plt.show()
```

Annexe

Code Python et Courbes : $2^{ème}$ Simulation

```python
import numpy as np
from matplotlib import pyplot as plt
from scipy.integrate import odeint

##Valeurs de la simulation R0>1##

u1=0
u2=0
u3=0

##Valeurs de la simulation R0<1##

u1=0.88
u2=0.7
u3=0.65

##

def Evolution(compartiments,t,beta,gamma,mu):
    S=compartiments[0]
    I=compartiments[1]
    R=compartiments[2]
    M=compartiments[3]
    dSdt=-(1-u1)*beta*S*I
    dIdt=(1-u1)*beta*S*I-(1-u3)*mu*I-(1+u2)*gamma*I
    dRdt=(1+u2)*gamma*I
    dMdt=(1-u3)*mu*I
    return [dSdt,dIdt,dRdt,dMdt]

t=np.linspace(0,15000,1000000)
compartiments=[100000,10000,0,0]

s=odeint(Evolution,compartiments,t,(7e-08,5e-04,2e-04))

plt.plot(t,s[:,0],label="Nombre de susceptibles")
plt.plot(t,s[:,1],label="Nombre d'infectés")
plt.plot(t,s[:,2],label="Nombre de rétablis")
plt.plot(t,s[:,3],label="Nombre de morts")
plt.legend()
plt.grid()
plt.show()
```