INSA TOULOUSE

PROJET DE MODÉLISATION

ETUDE ÉPIDÉMIOLOGIQUE À PARTIR DES MODÈLES SIS ET SIR STOCHASTIQUES

LELIÈVRE Hugo
GRENOUILLAT Alexia

3A Spécialité Mathématiques Appliquées

Année universitaire 2020/2021

Encadrant : HUANG Lorick

Table des matières

1 Introduction

Lorsque l'on s'intéresse à la propagation et au développement d'une épidémie, on s'aperçoit que de nombreux modèles ont été développés. Du déterministe au probabiliste, en passant par le stochastique, ces modèles visent à étudier de la manière la plus précise possible l'expansion d'une maladie infectieuse au sein d'une population. En analysant rapidement ces modèles, on se rend compte du rôle crucial que joue le paramètre R_0, qui décrit le nombre moyen de nouvelles infections dues à un individu malade. Néanmoins, les modèles classiques possèdent tous leurs limites, et R_0 est loin de décrire à lui seul la propagation d'une épidémie au sein d'une population réelle. Afin d'étudier l'évolution d'une épidémie selon les modèles SIR et SIS, à travers nos premières documentations, nous avons trouvé énormément d'approches temporelles du modèle reposant sur la résolution d'équations différentielles. Nous avions envie de partir sur quelque chose de plus "innovant" ; c'est là que nous est venue l'idée d'une représentation spatiale en modélisant la propagation d'une épidémie par un automate cellulaire.

Nous nous attacherons dans un premier temps à faire une très rapide description du modèle déterministe le plus classique, qui repose sur la résolution d'équations différentielles. Dans un second temps, nous vous présenterons notre approche numérique du modèle SIR par le biais de notre automate cellulaire. Nous chercherons notamment à établir un lien entre la probabilité p1 de contamination au contact d'un individu malade, et une probabilité p2 de rester malade entre un instant T et un instant T+1. Nous rapprocherons en particulier nos résultats avec les courbes obtenues par l'approche déterministe présentée précédemment. Enfin, nous aborderons une approche plus probabiliste avec l'étude des chaînes de Markov appliquée au modèle SIS – en travaillant toujours sur notre automate cellulaire. Notre objectif sera de montrer que, pour une matrice 2x2 dans n'importe quel état initial, la probabilité d'arriver à l'état absorbant tend vers 1 quand le temps devient infini. Pour conclure, nous évoquerons les limites de notre automate cellulaire ainsi que les pistes de poursuite d'étude que nous avions envisagées.

2 Présentation du modèle déterministe : la modélisation d'une épidémie par la résolution d'équations différentielles

Nous nous intéresserons plus particulièrement au modèle SIR, ie au cas d'une maladie infectieuse qui procure une immunité une fois l'individu guéri. Afin d'aborder cette approche, il est nécessaire de diviser notre population en trois catégories. Chaque individu de la population peut être dans 3 états :

$\rightarrow S(t)$: Sain et susceptible de contracter la maladie.

$\rightarrow I(t)$: Malade et répandant l'épidémie au contact des individus sains.

$\rightarrow R(t)$: Guéri et immunisé – en particulier, ne peut plus répandre la maladie –.

Nous pouvons de plus émettre les hypothèses suivantes :

$\rightarrow$ La taille de la population est suffisamment importante pour pouvoir supposer que S(t), I(t), R(t) sont des fonctions dérivables

$\rightarrow$ Un individu de la population considérée rencontre en moyenne β individus par unité de temps

$\rightarrow$ Le taux d'individus infectés qui deviennent guéris par unité de temps est donné par γI (où γ représente le taux de guérison).

Nous obtenons alors le système d'équations différentielles suivant :

$$\begin{cases} \frac{dS}{dt} &= -\frac{\beta IS}{N} \\ \frac{dI}{dt} &= \frac{\beta IS}{N} - \gamma I \\ \frac{dR}{dt} &= \gamma I \end{cases}$$

Ce système d'équation définit un modèle déterministe ; un tel modèle peut être défini comme étant un système qui obéit à des lois d'évolution non-probabilistes. Généralement, l'évolution de ce modèle au cours du temps est décrite par un sytème d'équations (le plus souvent, différentielles). Dans un système déterministe, on est capable de savoir ce qu'il se passe à chaque instant t, sans qu'aucune intervention probabiliste ne soit possible. Cette définition est contraire à celle d'un modèle stochastique, qui représente l'évolution en temps discret - dans notre cas - d'une variable aléatoire ; la dimension probabiliste y est donc omniprésente.

Ces équations différentielles nous permettent alors de déterminer le fameux paramètre R_0, que l'on définit rigoureusement comme le nombre d'individus infectés par une personne infectieuse placée dans une population d'individus susceptibles. Dans notre description du modèle déterministe, afin d'être en adéquation avec les conditions initiales de notre automate, nous considérons que le nombre d'individus susceptibles est très supérieur au nombre d'individus infectés au temps initial $t = 0$.

On sait qu'un individu malade infecte β individus par unité de temps. Afin de calculer le R_0, on a besoin de calculer la période moyenne d'infection. Comme le taux de guérison vaut γ, on en

déduit que la période moyenne d'infection est égale à $\frac{1}{\gamma}$. De plus, comme une personne infectieuse contamine en moyenne β individus par unité de temps, on obtient qu'une personne infectieuse infecte $\frac{\beta}{\gamma}$ individus sains. Donc, par définition de R_0, on arrive à la relation $R_0 = \frac{\beta}{\gamma}$.

Le principal avantage de R_0 est que sa valeur permet à elle seule de savoir si une épidémie va s'éteindre ou si elle va au contraire s'étendre. En effet, la maladie s'éteint si le nombre d'infectés diminue, autrement dit si la dérivée de la proportion des individus infectés est négative,

ie si $\frac{dI}{dt} < 0$

si $\frac{\beta I_0 S_0}{N} - \gamma I_0 < 0 \qquad$ avec $S_0 = N$ et $I_0 = I(t = 0)$

si $R_0 = \frac{\beta}{\gamma} < 1$

Dans le cas contraire, l'épidémie va se propager si $R_0 > 1$, ce qui correspond bien au fait que le nombre d'infectés augmente (car la dérivée de la proportion des individus malades est positive).

Nous allons maintenant vous présenter l'automate cellulaire probabiliste que nous avons mis au point dans le but d'étudier le modèle SIR.

3 Elaboration de l'automate cellulaire probabiliste appliqué au modèle SIR

3.1 Codage de l'automate

Nous avons fait le choix d'utiliser un automate cellulaire afin de visualiser l'évolution en temps réel de l'épidémie, avec la possibilité de détailler pour chaque étape de la propagation les 3 populations $S(t), I(t)$ et $R(t)$. Pour l'exploitation de notre automate, nous avons considéré qu'une étape équivalait à un jour. Nous avons ainsi construit un modèle simplifié, pour nous permettre de tester l'automate initial et de nous assurer de sa pertinence. Ce premier automate est composé des paramètres suivants :

→ Une grille de 40 cellules par 40 (1600 cellules au total, toutes de taille 15, modifiable selon les besoins de l'utilisateur)

→ Un bouton « Start » pour lancer l'automate cellulaire

→ Un bouton « Stop » pour mettre l'automate sur pause

→ Un bouton « Exit » pour arrêter l'automate et afficher le graphique final

→ Un bouton « Step » pour suivre l'évolution de la propagation de l'épidémie jour par jour

On définit la taille de notre tableau (ici 40x40), les probabilités qui vont régir l'évolution de notre automate, et enfin quelques variables qui nous serviront à tracer le graphique de l'évolution des populations saines, malades et guéries en fonction du temps

[21]:
```
# Dimension du monde
NbL = 40  # hauteur du tableau
NbC = 40  # largeur du tableau
a = 15   # taille d'une cellule

#probabilités de changement
p1=0.5 #proba d'être infecté en contact d'un malade
p2=0.9 #proba de rester malade

#affichage de la courbe des infectés et des sains
L=[]
temps=[]
nb_passages=0
malades_vivants=True
```

Afin de visualiser l'évolution de la maladie, il est nécessaire de créer une interface graphique, où on a ajouté un certain nombre de fonctionnalités dans le code ci-dessous. On a introduit une va-

riabe globale flag qui nous permet de stopper la progression de la maladie et de mettre l'automate sur pause.

[ ]:
```
voir annexe 2
```

A présent, on crée la variable tab qui sera notre matrice contenant les informations concernant chaque cellule : 0 pour sain, 1 pour malade et 2 pour guéri. La matrice cell va quant à elle nous permettre de créer la grille qui va représenter tab.

[ ]:
```
voir annexe 3
```

On implémente la fonction qui permet de relier tab à cell ; elle permet de changer de couleur les cases de la grille représentée par cell en fonction des valeurs de la matrice tab.

[ ]:
```
voir annexe 4
```

Une fois l'automate terminé, nous avons également défini les règles de propagation de l'épidémie : Chaque cellule représente un individu lambda et possède 8 voisins (2 latéraux, 1 au-dessus, 1 en-dessous, 4 en diagonales). Une cellule peut être dans 3 états :

$\rightarrow$ Saine, faisant donc partie de la proportion $S(t)$ de la population. Elle est laissée en blanc sur l'interface graphique

$\rightarrow$ Malade, faisant donc partie de la proportion $I(t)$ de la population. Elle est représentée par une case rouge sur l'interface graphique

$\rightarrow$ Guérie, faisant donc partie de la proportion $R(t)$ de la population. Elle est représentée par une case bleue sur l'interface graphique

Chaque cellule malade peut contaminer ses 8 cellules voisines (et aucune autre). Toutes les cellules voisines d'une cellule malade ont la même probabilité d'être infectées (aucun individu n'est ici considéré comme « à risque » puisque nous avons commencé par un modèle extrêmement simplifié). De plus, il faut préciser que le nombre de voisins malades d'un individu n'influence pas ses chances d'être contaminé ; on distingue juste le cas où l'individu a au moins un voisin malade du cas où il n'en a aucun (toujours dans un souci de simplicité du modèle).

Au niveau de l'automate, nous avons précédemment ajouté une fonction clic gauche qui nous

permet de placer un ou plusieurs malades en cliquant sur la grille ; cependant, dans une optique d'étude de notre automate, nous avons décidé de placer les 3 mêmes malades au départ de chaque simulation. En effet, le nombre de malades à l'instant 0 et leurs positions étant sûrement des facteurs dans le développement de l'épidémie, nous avons donc choisi cette méthode. De plus, avec un seul point de départ, le processus étant aléatoire, on s'expose à une extinction immédiate de la maladie. Le choix de 3 points de départs assez éloignés nous semblait un bon compromis pour éviter ce phénomène, tout en ne plaçant pas trop de malades au temps 0.

[27]:
```
#on place les premiers malades
def initialiser_malades():
    global tab
    tab[12,31]=1
    tab[25,15]=1
    tab[37,13]=1
```

Nous avons ensuite codé la fonction d'évolution de la grille ; il a fallu faire attention à ne pas modifier directement la matrice tab et à utiliser une copie temporaire. En effet, suivant l'ordre de parcours de la matrice tab, on aurait pu provoquer des effets à la chaine. Par exemple, si on considère une matrice initiallement nulle avec seulement tab[0,0]=1 on a tab[0,1] qui est voisin et peut donc être infecté. Si on affecte 1 à tab[0,1] immédiatement, alors dans le parcours de la matrice, tab[0,2] sera voisin de tab[0,1] qui est à présent malade. Bien que tab[0,2] n'avait aucun voisin malade au jour précédent il pourrait le devenir ici, ce qui est incohérent. C'est pour cela que nous avons décidé de créer une deuxieme matrice dans laquelle on met les valeurs de la matrice tab au temps t+1 obtenues à partir de la matrice tab au temps t.

[28]:
```
# Calcul de la prochaine génération
def evolution_grille():
    global tab
    tab2=np.zeros((NbL,NbC))
    for num_ligne in range(NbL):
        for num_colonne in range(NbC):
            if tab[num_ligne,num_colonne]==0:
                voisin_malade=False
```

```
                if tab[(num_ligne-1)%NbL,num_colonne]==1 or␣
↪tab[num_ligne,(num_colonne-1)%NbC]==1 or tab[(num_ligne+1)%NbL,num_colonne]==1␣
↪or tab[num_ligne,(num_colonne+1)%NbL]==1 or␣
↪tab[(num_ligne-1)%NbL,(num_colonne-1)%NbC]==1 or␣
↪tab[(num_ligne+1)%NbC,(num_colonne-1)%NbC]==1 or␣
↪tab[(num_ligne-1)%NbC,(num_colonne+1)%NbC]==1 or␣
↪tab[(num_ligne+1)%NbC,(num_colonne+1)%NbC]==1 :
                    voisin_malade=True
                if voisin_malade:
                    temp=uniform(0,1)
                    if temp < p1:
                        tab2[num_ligne,num_colonne]=1
            elif tab[num_ligne,num_colonne]==1:
                temp=uniform(0,1)
                if temp < p2:
                    tab2[num_ligne,num_colonne]=1
                else:
                    tab2[num_ligne,num_colonne]=2
            else:
                tab2[num_ligne,num_colonne]=2
    tab=tab2.copy()
```

Afin d'avoir un graphique représentant l'évolution des populations en fonction du temps, nous avons implémenté ces fonctions permettant de compter le nombre de cases dans chaque état et de les ajouter à L.

[ ]:
```
voir annexe 5
```

[29]:
```
#pour faire evoluer la grille et enregistrer dans L le compte de chaque␣
 ↪population dans le nouvel etat de la grille
def appliquer_regles():
    evolution_grille()
    ajouter_graphique()
```

On crée ensuite une fonction itérer qui nous permet d'appliquer les règles de notre automate et de passer d'un état au temps t de la grille à l'état suivant au temps t+1.

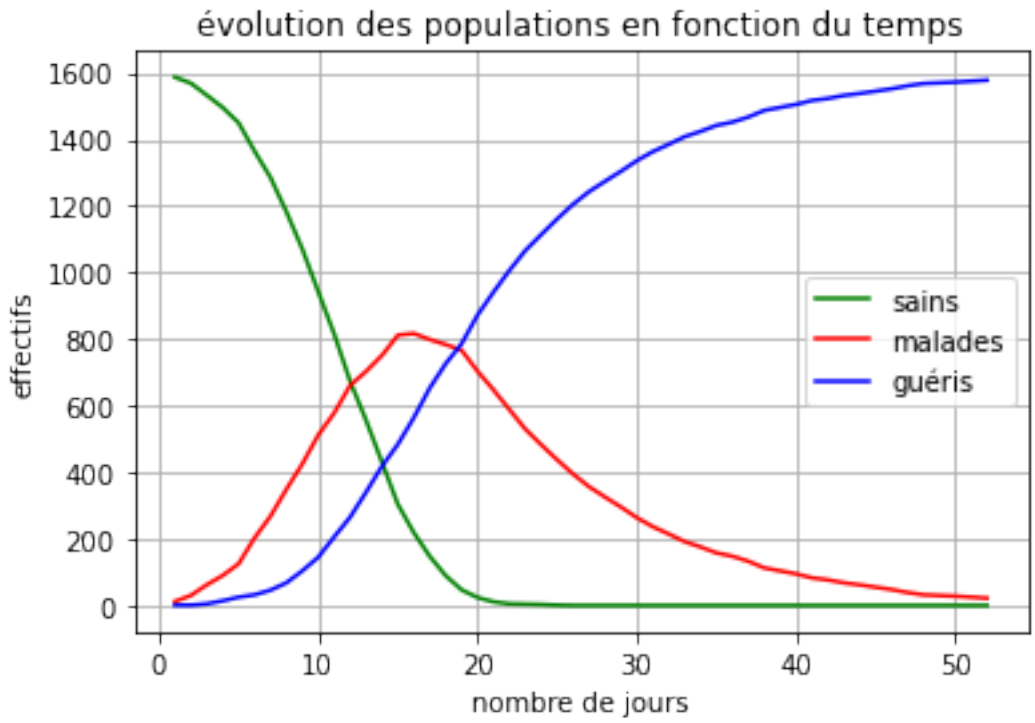
[30]:
```
# Calculer et dessiner la prochaine génération
def iterer():
    global flag
    appliquer_regles()
    dessiner()
    if not malades_vivants:
        flag=0
    if flag==1:
        fenetre.after(1, iterer)
    else:
        flag=0
```

On peut à présent lancer l'automate en initialisant flag à 0 et en executant les fonctions suivantes :

[ ]:
```
voir annexe 6
```

Et enfin on affiche le graphe de l'évolution des populations :

[32]:
```
voir annexe 7
```



3.2 Analyse de l'influence de p1 et p2 sur la propagation de l'épidémie

Une fois cette étape de création et d'initialisation de l'automate cellulaire, nous nous sommes ensuite intéressés aux probabilités suivantes :

→ La probabilité pour une cellule voisine d'une cellule malade d'être contaminée, représentée par p1 dans notre code ;

→ La probabilité pour une cellule malade de rester malade le jour suivant, représentée par p2 dans notre code ; Notre objectif était de trouver une relation entre ces deux probabilités. Pour cela, nous nous sommes focalisés sur la question suivante : à partir de quel(s) critère(s) peut-on considérer que l'épidémie s'arrête naturellement ? Nous avons alors choisi des probabilités aléatoires (mais tout de même un minimum réalistes) pour p1 et p2, et nous avons constaté deux choses :

→ Dans certains cas, l'épidémie se propage jusqu'à contaminer un nombre important de cellules mais ne peut pas s'étendre davantage. En effet, certaines « régions » de la grille lui sont devenues inaccessibles (c'est-à-dire qu'un bloc de cellules saines est entouré de cellules guéries, les rendant inaccessibles pour la maladie). Cela se rapproche d'un phénomène de percolation ; néamnmoins, l'univers de notre automate cellulaire étant tourique, on ne peut pas vraiment faire d'analogie entre nos observations et ce phènomène (puisque la percolation nécessite la présence de bords). L'épidémie s'arrête donc, mais de manière non naturelle (en effet, si la population se brassait,

l'épidémie pourrait redémarrer). Dans ce cas, nous avons considéré que l'épidémie ne s'éteignait pas « naturellement ».

$\rightarrow$ Dans d'autres, au contraire, l'épidémie se propage de manière plus ou moins importante et finit par s'éteindre alors qu'elle aurait pu continuer à se diffuser (c'est le cas lorsque les cellules malades ne contaminent plus aucune cellule voisine avant d'être guéries). Dans ce cas, nous avons considéré que l'épidémie s'éteignait naturellement.

Afin de trouver une potentielle relation entre p1 et p2, nous avons opté pour une approche expérimentale ; nous sommes partis de probabilités relativement réalistes (p1=0.1 et p2=0.5) et avons réalisé une dizaine de tests en partant à chaque fois avec les conditions initiales suivantes :

$\rightarrow S(0)$=1597

$\rightarrow I(0)$=3

$\rightarrow R(0)$=0

Pour chaque test, nous avons déterminé si l'épidémie s'arrêtait naturellement ou non. Nous nous sommes également intéressés à d'autres données qui nous paraissaient pertinentes, puis nous avons dressé un tableau de ce style :

$NumSimulation$	$ExtinctionNaturelle?$	$NbJours$	$NbTotalInfects$
1	1	10	10
2	1	10	15
3	1	5	11
4	1	5	7
5	1	9	15
6	1	14	18
7	1	32	59
8	1	29	31
9	1	7	13
10	1	8	8

Arrivés à ce stade, il nous a ensuite fallu réussir à exprimer p2 en fonction de p1 ; une relation existe-t-elle ? Si oui, de quelle forme est-elle ? Linéaire ? Affine ? Polynomiale ? Exponentielle ? Pour la déterminer, nous avons essayé de trouver, pour 9 valeurs de p2, les valeurs de p1 pour lesquelles, sur 10 tests, il y avait autant de cas où l'épidémie s'arrêtait naturellement que de cas où son arrêt n'était pas naturel. Puisque, dans le cas p1=0.1 et p2=0.5, l'épidémie s'arrête naturellement dans 100% des cas, nous avons augmenté p1 à 0.2. et avons obtenu les données suivantes :

$NumSimulation$	$ExtinctionNaturelle?$	$NbJours$	$NbTotalInfects$
1	0	66	1254
2	0	73	1256
3	0	67	1316
4	0	73	1319
5	0	62	1188
6	0	77	1235
7	0	61	1335
8	0	104	1202
9	0	74	1284
10	0	60	1288

Cette fois, pour la totalité des simulations, l'épidémie ne s'arrête jamais naturellement. Nous avons donc procédé par dichotomie (en gardant 2 décimales de précision) et avons trouvé la valeur « optimale de p1 » : p1=0.18 :

$NumSimulation$	$ExtinctionNaturelle?$	$NbJours$	$NbTotalInfects$
1	0	77	1157
2	0	71	1145
3	1	51	248
4	1	71	990
5	1	44	414
6	1	95	1059
7	1	22	47
8	0	78	1110
9	0	87	1059
10	0	123	1100

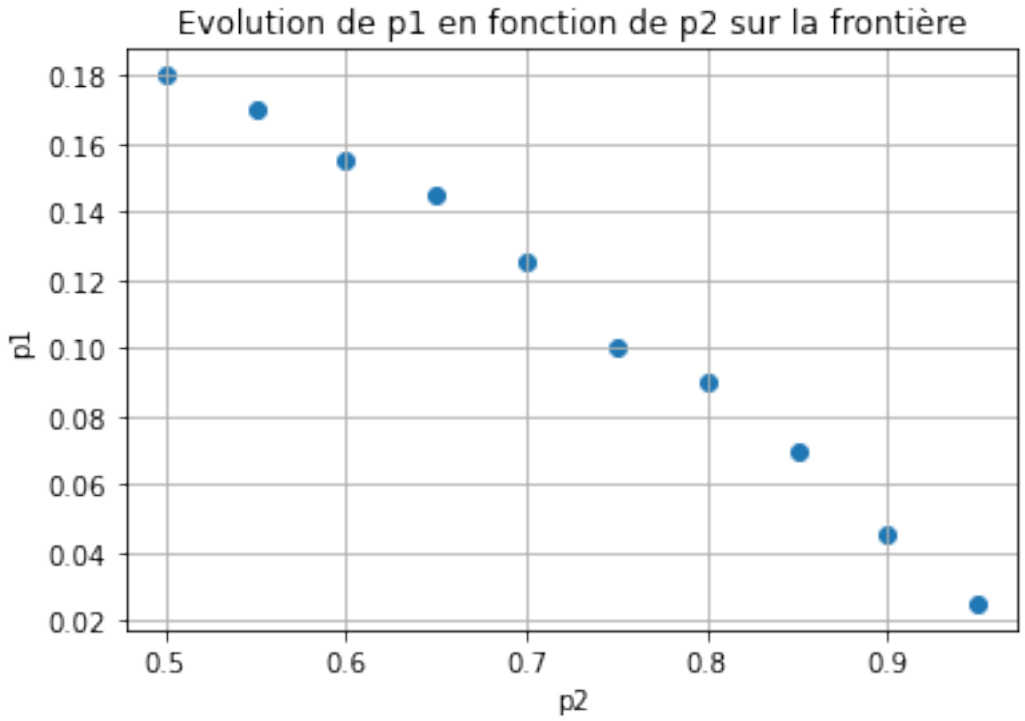
Nous avons ensuite augmenté p2 jusqu'à 0.95 (avec un pas de 0.05) en procédant toujours de la façon décrite ci-dessus. Nous avons ainsi trouvé p1 et les p2 suivants :

$p1$	$p2$
0.18	0.50
0.17	0.55
0.155	0.60
0.145	0.65
0.13	0.70
0.10	0.75
0.09	0.80
0.07	0.85
0.045	0.90
0.025	0.95

Afin de visualiser graphiquement ces résultats, nous avons tracé un nuage de points de p1 en fonction de p2 :

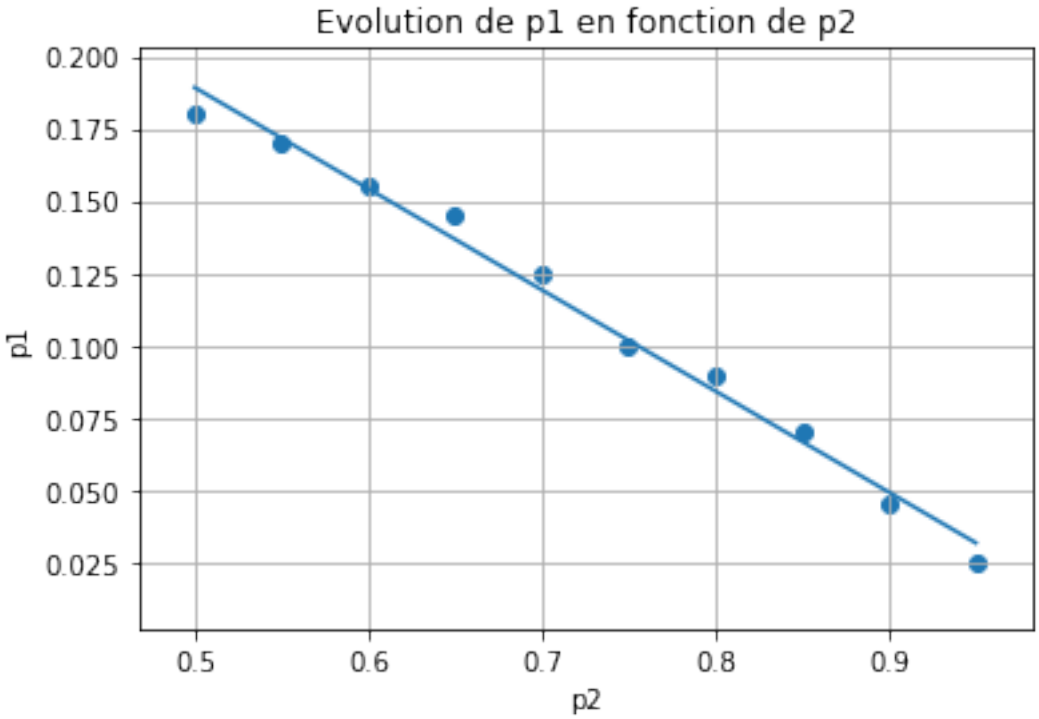
[12]:
```
liste_p1_frontiere=[0.18,0.17,0.155,0.145,0.125,0.1,0.09,0.07,0.045,0.025]
liste_p2_frontiere=[0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9,0.95]

plt.scatter(liste_p2_frontiere,liste_p1_frontiere)
```

[12]:
```
<matplotlib.collections.PathCollection at 0x24219159988>
```



D'un point de vue purement visuel, les points semblent presque s'aligner selon une droite. Pour confirmer cette impression, nous avons tracé la droite de régression linéaire de ces points :

[35]:
```
voir annexe 8
```




```
coeff directeur =  -0.3496969696969686
ordonnée à l'origine =  0.3640303030303023
```

Cela confirme donc bien notre ressenti ; p1 et p2 sont reliées de manière affine :

→ Coeff directeur = -0.35

→ Ordonnée à l'origine = 0.364

On obtient donc p1 = -0.35 * p2 + 0.364

Cette droite représente la frontière entre extinction naturelle ou non : au-dessus de la droite, l'épidémie ne s'éteint pas naturellement ; en-dessous, l'extinction est naturelle. Cette frontière est défi-

nie par un équilibre entre les probabilités p1 de contamination et p2 de rester malade (avec 1-p2 proba de guérison). Il est donc logique que cette droite soit décroissante : en effet, pour rester sur la frontière quand le taux de guérison augmente (ie 1-p2 augmente, donc p2 diminue), il faut que le taux de contamination augmente en même temps (ie p1 augmente). En appliquant ce raisonnement au graphique, on obtient bien que p1 doit diminuer quand p2 augmente.

Notre automate cellulaire étant aléatoire, il nous a paru intéressant de calculer les probabilités exactes de changement d'état. Nous allons donc à présent considérer la grille d'évolution comme une chaîne de Markov afin d'essayer de déterminer complètement son comportement.

4 Approche chaîne de Markov sur le modèle SIS

4.1 Caractérisation de la chaîne de Markov : recherche de la matrice de transition

Le fonctionnement de notre automate cellulaire est clairement basé sur un processus markovien. En effet, à un instant T donné, l'état futur de chaque cellule dépend uniquement de son état présent et de celui de ses 8 voisins. On peut dès lors écrire les deux propriétés de Markov, dont on se servira par la suite : Soit $(X_n)_{n\in\mathbb{N}}$ une suite de v.a. à valeurs dans un espace d'état E = 0,1 (dans notre cas, chaque suite (X_n) représente l'état d'un individu). Si cette suite est une chaîne de Markov, alors pour tous n *in* $\mathbb{N}$ et $x_0, x_1, ..., x_{n-1}, x, y \in E$:

$\rightarrow$ (La propriété de Markov) : $\mathbb{P}(X_{n+1} = y | X_n = x; X_{n-1} = x_{n-1}; ...; X_0 = x_0) = \mathbb{P}(X_{n+1} = y | X_n = x)$.

$\rightarrow$ (L'homogénéité) : La probabilité $\mathbb{P}(X_{n+1} = y | X_n = x)$ ne dépend pas de n. Dans ce cas on note $\mathbb{P}(x, y) = \mathbb{P}(X_{n+1} = y | X_n = x)$: c'est la probabilité de transition de l'état x à l'état y.

En reformulant, selon la propriété de Markov, si :

$\rightarrow$ X_n représente le présent

$\rightarrow$ les X_k le passé lorsque $k \in \{0, ..., n-1\}$, et le futur quand $k \geq n+1$

Alors "le futur est indépendant du passé, conditionnellement au présent". Par ailleurs, la loi de X_0 est appelée loi initiale de la chaîne.

Pour simplifier, on commence par une matrice de taille 2x2 ; en effet, il existe seulement un état absorbant pour $2^4 = 16$ états possibles. Chacune des variables de la matrice (qu'on considérera comme un individu, pouvant être sain ou malade) peut prendre 2 valeurs distinctes : 0 ou 1 (correspondant respectivement à sain et malade).

Ces 16 états sont :

$$1 : \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad 2 : \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad 3 : \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad 4 : \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad 5 : \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad 6 : \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

$$7 : \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \quad 8 : \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad 9 : \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad 10 : \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \quad 11 : \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \quad 12 : \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

$$13 : \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad 14 : \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad 15 : \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \quad 16 : \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

On implémente ces 16 états en python (voir annexe) :

[ ]:
```
voir annexe 9 :
```

Nous avons donc cherché à déterminer sa matrice de transition. Pour cela, intéressons nous à un individu de la matrice. Si cet individu est malade, il va rester malade le jour d'après avec une probabilité p2, et devenir sain au jour d'après avec une probabilité 1-p2. S'il est sain on peut différencier 2 cas :

$\rightarrow$ soit l'individu n'a pas de voisin malade, et alors il restera sain au jour d'après avec une probabilité 1 ;

$\rightarrow$ soit il possède au moins un voisin malade et il deviendra malade au jour suivant avec une probabilité p1 ;

Pour réaliser des calculs par la suite, on notera un individu malade : $\begin{pmatrix} 0 & 1 \end{pmatrix}$ et un individu sain : $\begin{pmatrix} 1 & 0 \end{pmatrix}$

S'il est sain avec uniquement des voisins sains, la matrice de transition est l'identité. En effet, il restera sain à coup sûr puisqu'il n'a pas de voisin malade.

Sinon, on peut donc écrire la matrice de transition :

$$\begin{pmatrix} 1-p1 & p1 \\ 1-p2 & p2 \end{pmatrix}$$

[13]:
```
M=np.array([[1-p1,p1],[1-p2,p2]])
```

On définit aussi les nouvelles dimensions des matrices (qui sont de taille 2x2) :

[14]:
```
NbL2=2
NbC2=2
```

On implémente des fonctions permettant d'écrire les indidus sous forme de vecteurs comme décrit ci-dessus :

malade : $\begin{pmatrix} 0 & 1 \end{pmatrix}$ et sain : $\begin{pmatrix} 1 & 0 \end{pmatrix}$

[ ]:
```
voir annexe 10
```

Maintenant que les individus sont sous forme de vecteurs, on peut appliquer à chaque individu la matrice de transition adéquate en fonction de leur état et de leur voisinage :

→ Si l'individu est sain avec uniquement des voisins sains, il reste sain à 100%, donc on lui applique la matrice de transition I_2

→ Si ce n'est pas le cas, on lui applique la matrice : $\begin{pmatrix} 1-p1 & p1 \\ 1-p2 & p2 \end{pmatrix}$

[15]:
```
def appliquer_M(X,M):
    Xinit=np.copy(X)
    X=transfo_X(X)
    X2=X.copy()
    for num_ligne in range(NbL2):
        for num_colonne in range(NbC2):
            if Xinit[(num_ligne-1)%NbL2,num_colonne]==1 or␣
 ↪Xinit[num_ligne,(num_colonne-1)%NbC2]==1 or␣
 ↪Xinit[(num_ligne+1)%NbL2,num_colonne]==1 or␣
 ↪Xinit[num_ligne,(num_colonne+1)%NbL2]==1 or␣
 ↪Xinit[(num_ligne-1)%NbL2,(num_colonne-1)%NbC2]==1 or␣
 ↪Xinit[(num_ligne+1)%NbC2,(num_colonne-1)%NbC2]==1 or␣
 ↪Xinit[(num_ligne-1)%NbC2,(num_colonne+1)%NbC2]==1 or␣
 ↪Xinit[(num_ligne+1)%NbC2,(num_colonne+1)%NbC2]==1 :
                new_Y=np.dot(X[num_ligne,num_colonne],M)
                X2[num_ligne,num_colonne]=new_Y.copy()
            else:
                if Xinit[num_ligne,num_colonne]==0:
                    X2[num_ligne,num_colonne]=np.array([1,0])
                else:
                    new_Y=np.dot(X[num_ligne,num_colonne],M)
                    X2[num_ligne,num_colonne]=new_Y.copy()
    return X2
```

On peut à présent calculer la probabilité que chaque individu de la matrice Xinit se retrouve dans l'état de la matrice Xfinal au jour suivant. On obtient ainsi une matrice contenant la probabilité que chaque individu passe dans l'état d'arrivée voulu.

[16]:
```
def matrice_proba_changement(Xinit,Xfinal,M):
    X=appliquer_M(Xinit,M)
```

```
    res=np.zeros((NbL,NbC))
    for num_ligne in range(NbL2):
        for num_colonne in range(NbC2):
            if Xfinal[num_ligne,num_colonne]:
                res[num_ligne,num_colonne]=X[num_ligne,num_colonne][1]
            else:
                res[num_ligne,num_colonne]=X[num_ligne,num_colonne][0]
    return res
```

L'événement "passer de $X_{initial}$ à X_{final}" ou "$(X_{n+1} = X_{final}|X_n = X_{initial})$" correspond à l'intersection des événements sur tous les individus Y_i de la matrice X_n (qui contient N individus) : " l'individu Y_i est passé de l'état $Y_{i,initial}$, dans X_{init} à l'état $Y_{i,final}$ dans X_{final}",

soit "$(\bigcap_{i=0}^{N}(Y_{i,n+1} = Y_{i,final}|X_n = X_{initial}))$ ".

$\mathbb{P}(X_{n+1} = X_{final}|X_n = X_{initial})$

$= \mathbb{P}(\bigcap_{i=0}^{n}(Y_{i,n+1} = Y_{i,final}|X_n = X_{initial}))$

$= \prod_{i=1}^{n} \mathbb{P}(Y_{i,n+1} = Y_{i,final}|X_n = X_{initial}))$

car les $(Y_{i,n+1})$ $1 \leq i \leq N$ sont mutuellement indépendants, ils ne dépendent que de X_n

$= \prod_{i=1}^{n} \mathbb{P}(Y_{i,n+1} = Y_{i,final}|$ voisinage de $Y_{i,n}))$

La probabilité de passer de X_{init} à X_{final} correspond donc au produit des probabilités de changement de chaque individu, c'est-à-dire le produit de tous les coefficients de la matrice obtenue précédemment, soit matrice_proba_changement(Xinit,Xfinal,M).

[ ]:
```
voir annexe 11
```

On peut finalement calculer la matrice de transition globale de notre chaîne de Markov en calculant la probabilité de changement, partant de tous les états initiaux possibles vers tous les états finaux possibles (en sachant que les états initiaux et finaux possibles sont les mêmes) :

[17]:
```
def matrice_transition_globale(liste_X_possibles):
    res=np.zeros((16,16))
    for i in range(16):
        for j in range(16):
            Xinit=np.array(liste_X_possibles[i])
            Xfinal=np.array(liste_X_possibles[j])
```

```
            temp=afficher_proba_totale_changement(Xinit,Xfinal)
            res[i,j]=temp
    return res
```

On ajoute une petite fonction pour vérifier la cohérence de la matrice obtenue. On contrôle ainsi que la somme des coefficients sur chaque ligne vaut bien 1.

[ ]:
```
voir annexe 12
```

4.2 Etude de la convergence vers l'élément absorbant

Pour étudier le comportement de la chaîne de Markov en temps long, il nous suffit d'étudier le comportement de la matrice de transition lorsqu'on l'élève à une grande puissance. Pour cela, on commence par créer la fonction permettant d'élever la matrice de transition à la puissance n :

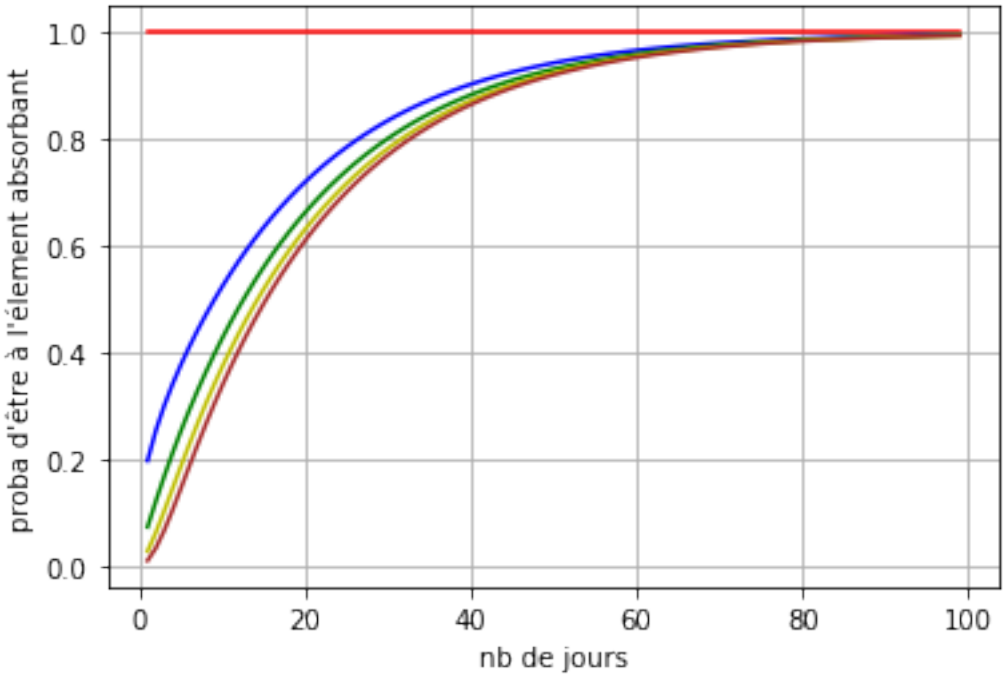
[ ]:
```
voir annexe 13
```

On cherche à présent la probabilité d'arriver à l'élement absorbant (à savoir la matrice nulle, ie tout le monde est sain), en fonction de l'état de départ $X_{initial}$. Pour cela, on élève la matrice de transition à la puissance n, puis on en extrait la première colonne.

[18]:
```
def proba_arriver_au_nul(n,P):
    liste_n=np.array([k for k in range(1,n)])
    liste_proba_nul=[]
    for n in liste_n:
        P_n=P_puissance_n(n,P)
        liste_proba_nul.append(P_n[:,0])
    res=np.array(liste_proba_nul)
    return res,liste_n
```

On peut alors tracer l'évolution de cette probabilité en fonction du nombre de jours :

[24]:
```
annexe 14
```



On remarque que la probabilité d'arriver au nul après n jours est la même pour tous les états initiaux contenant le même nombre de malades. En effet :

$\longrightarrow$ Si un seul individu de la matrice est malade, la probabilité de passer de $X_{initial}$ à $X_{absorbant}$ correspond au produit de 4 probabilités :

$\rightarrow$ La probabilité que l'individu devienne sain

$\rightarrow$ 3x(La probabilité qu'un individu sain avec un voisin malade reste sain)

Peu importe la position du malade, la probabilité de passer de $X_{initial}$ à $X_{absorbant}$ est donc la même.

$\longrightarrow$ Si deux individus de la matrice sont malades, la probabilité de passer de $X_{initial}$ à $X_{absorbant}$ correspond au produit de 4 probabilités :

$\rightarrow$ 2x(La probabilité qu'un individu malade devienne sain)

$\rightarrow$ 2x(La probabilité qu'un individu sain avec un voisin malade reste sain)

Peu importe la position des malades, la probabilité de passer de $X_{initial}$ à $X_{absorbant}$ est donc la même.

$\longrightarrow$ Si 3 ou 4 individus sont malades, le raisonnement est exactement le même.

Les états contenant uniquement un malade sont plus proches de l'élément absorbant (ie seul un malade doit guérir). Or, un malade reste malade avec une probabilité p2 = 0.8 et un individu sain reste sain avec une probabilité 1-p1 = 0.85. Il est donc d'autant plus facile d'arriver à l'état absorbant que le nombre d'individus qui doivent changer d'état est faible. Il est donc normal que la courbe bleue(correspondant à un seul malade initial) parte de plus haut que la courbe verte(ie deux malades initiaux), elle même plus haute que la courbe jaune (ie 3 malades), elle même plus haute que la courbe marron (ie 4 malades).

On peut donc voir sur le graphique que $\mathbb{P}(X_n = X_{absorbant}) \underset{n\to+\infty}{\longrightarrow} 1$

Ce qui est logique puisque $\mathbb{P}(X_{n+1} = X_{absorbant}|X_n = X_{initial}) > 0 \qquad \forall\, X_{initial}$.

On peut donc introduire $a > 0 \mid \forall\, X_{initial}$, $\mathbb{P}(X_{n+1} = X_{absorbant}|X_n = X_{initial}) > a$

On a alors $\mathbb{P}(X_{n+1} \neq X_{absorbant}|X_n = X_{initial}) < 1 - a < 1 \qquad \forall\, X_{initial}$

Puis $\mathbb{P}(X_{n+p} \neq X_{absorbant}|X_n = X_{initial}) < (1-a)^p \underset{p\to+\infty}{\longrightarrow} 0 \qquad \forall\, X_{initial}$

Enfin, on remarque que suivant les valeurs de p1 et de p2, la courbe va se rapprocher plus ou moins vite de 1. Puisque l'on arrive presque sûrement à l'état absorbant (ie tous les individus sont sains) quel que soit l'état de départ, on peut donc entrevoir la limite du modèle : l'épidémie finit toujours par s'arrêter. Cela s'explique en grande partie par la taille réduite de nos matrices de départ (2x2). Cependant, pour des raisons de temps de calcul et de complexité, nous ne pouvons pas étudier cette approche pour notre véritable automate. En effet, en taille 4, la matrice de transition est de taille $2^{16} = 65536$. Les calculs sous Python sont déjà irréalisables, autant dire qu'il n'est même pas envisageable de le faire en taille 40.

Les parties précédentes étant le fruit d'une conception personnelle, nous avons essayé de recontextualiser nos approches par rapport à la litterature déjà existante. En effet, il nous a paru à la fois utile et intéressant de comparer nos modèles et nos résultats avec des écrits scientifiquement certifiés afin de juger de la pertinence de nos analyses.

5 Mise en perspective de notre modèle avec la littérature relative aux automates cellulaires appliqués à une étude épidémiologique

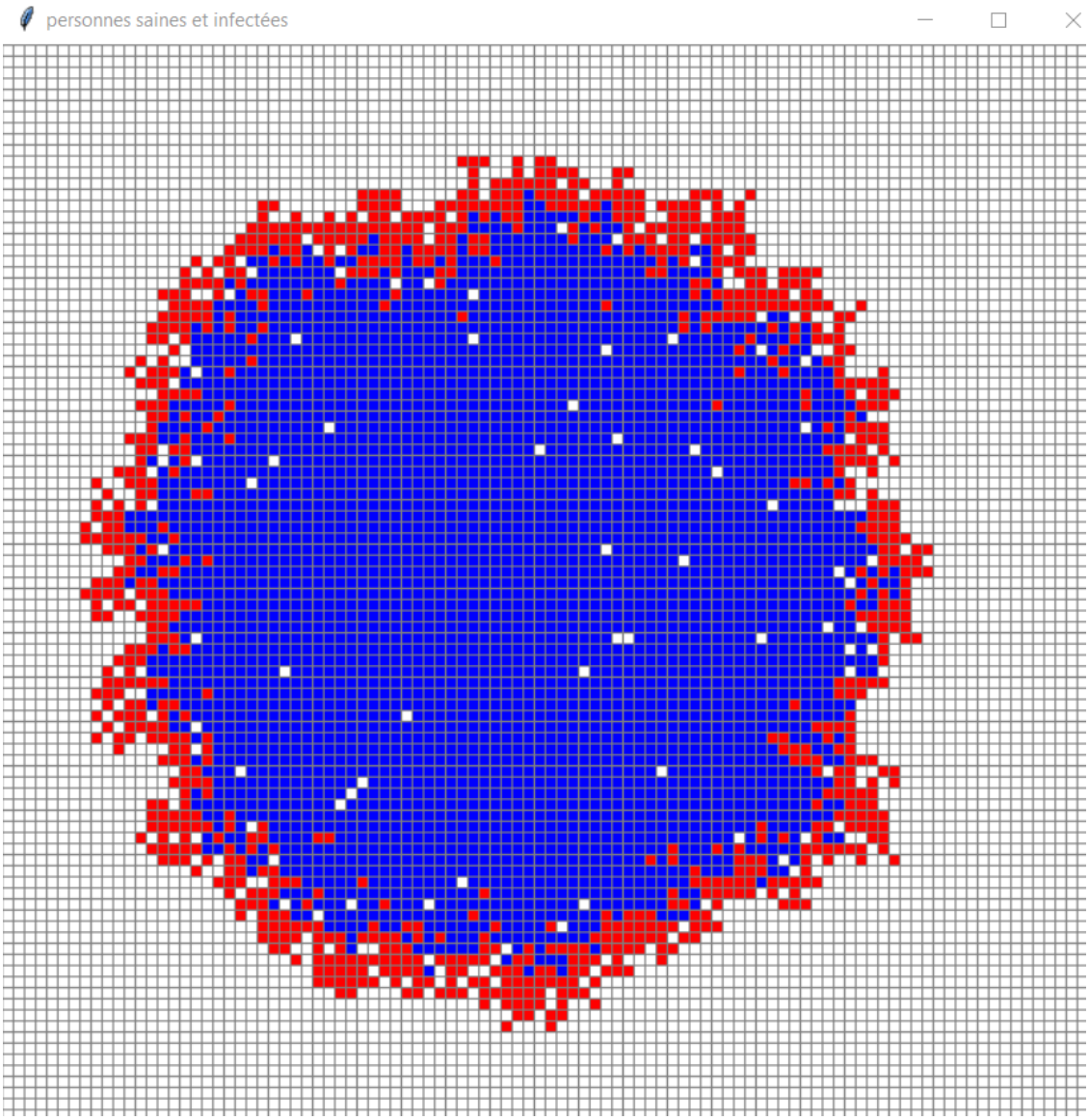
5.1 Comparaison entre notre automate et les autres automates présents dans la littérature

Au cours de nos recherches, nous avons été assez surpris de constater que l'approche automate cellulaire dans un cadre épidémiologique était relativement peu développée dans la littérature.

Dans un premier temps, après avoir conçu notre modèle, nous avons cherché s'il existait déjà dans la littérature un automate cellulaire régi par le même genre de règles afin de pouvoir vérifier la validité de nos travaux. Néanmoins, nous nous sommes très vite aperçus que la littérature épidémiologique traitait en grande majorité l'approche déterministe – et donc la résolution d'équations différentielles –, mais très peu de documents portaient sur l'utilisation d'automates cellulaires. De plus, la plupart des travaux suivant cette approche développe des règles d'évolution de la grille beaucoup plus complexes que les nôtres, avec des paramètres dont nous n'avons pas tenu compte par souci de simplicité. Il était donc assez délicat de comparer notre automate avec ceux qui existaient déjà ; néanmoins, nous avons pu remarquer plusieurs points communs intéressants entre les deux :

$\rightarrow$ La présence d'un « front » de propagation de l'épidémie, avec les individus guéris en bleu à l'intérieur du disque et les infectieux situés en périphérie, qui délimitent une sorte de frontière entre les populations $S(t)$ et $R(t)$.

[32]:
```
voir annexe 7
```



→ Un équilibre entre les différents paramètres impliqués : en réussissant à établir des relations mathématiques entre ces paramètres, on s'aperçoit qu'il existe, pour chacun d'entre eux, des valeurs limites au-delà desquelles l'épidémie aura tendance à exploser. De manière générale, nous avons remarqué que les probabilités de contamination et de guérison étaient liées par une relation linéaire ; à partir de là, il devient aisé de déterminer dans quelles circonstances une maladie va davantage se propager.

→ Le fait qu'une fois une sous-population d'individus sains est totalement encerclée par des cel-

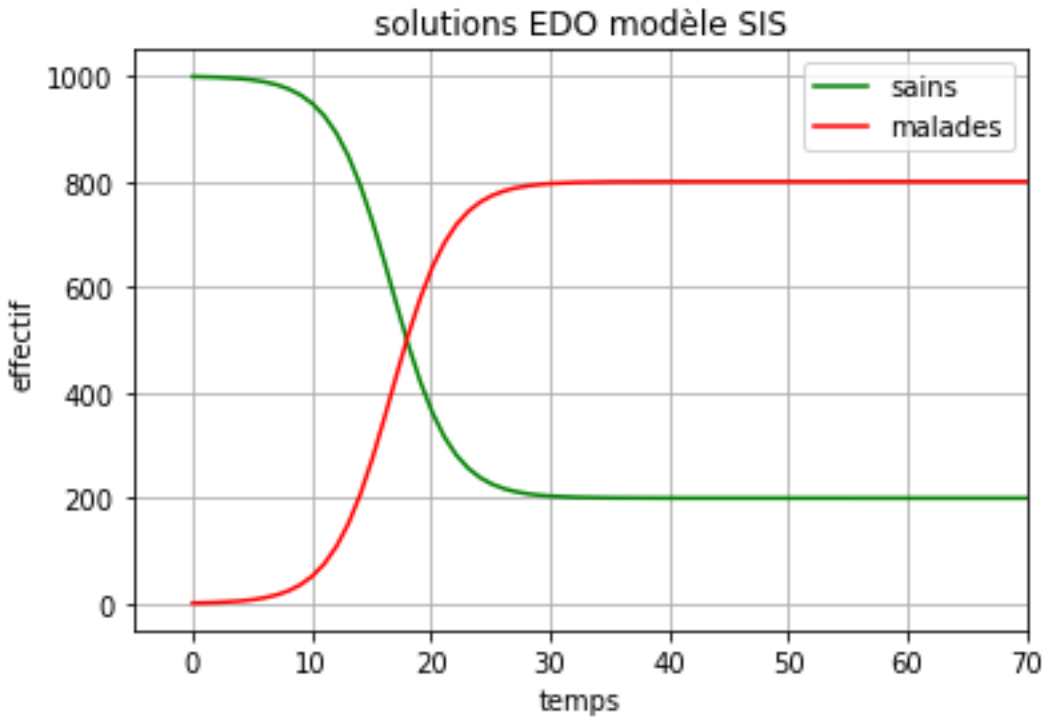
lules guéries, l'épidémie ne peut plus se propager parmi ce petit groupe (hormis si un cas infectieux se déclare aléatoirement au sein de cette sous-population, mais ce n'est pas le cas dans la plupart des configurations).

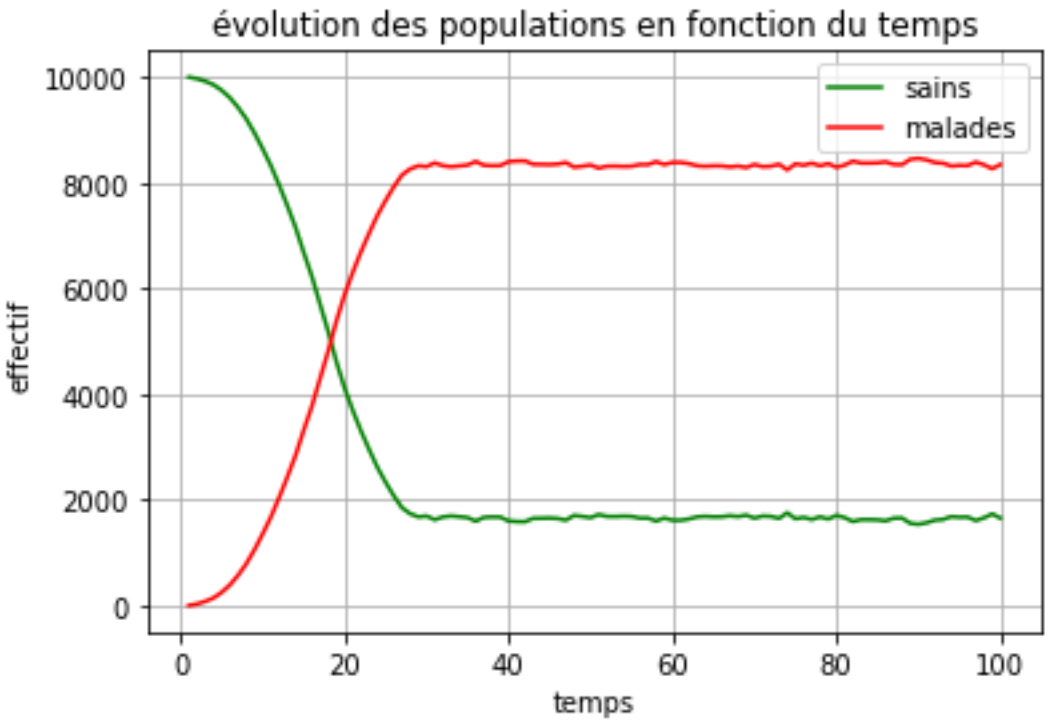
Enfin, nous avons cherché à comparer nos résultats avec ceux obtenus à partir de la résolution des équations différentielles.

5.2 Comparaison entre notre automate SIS et les solutions du systeme differentiel SIS

Tout d'abord nous avons voulu comparer nos résultats dans le cas du modèle SIS. Etant plus simple que le modèle SIR, nous pensions donc que les similitudes (ou les différences) seraient plus visibles. Pour cela, nous avons simplement modifié certaines lignes du code de notre automate SIR, et ainsi créé notre nouvel automate SIS.

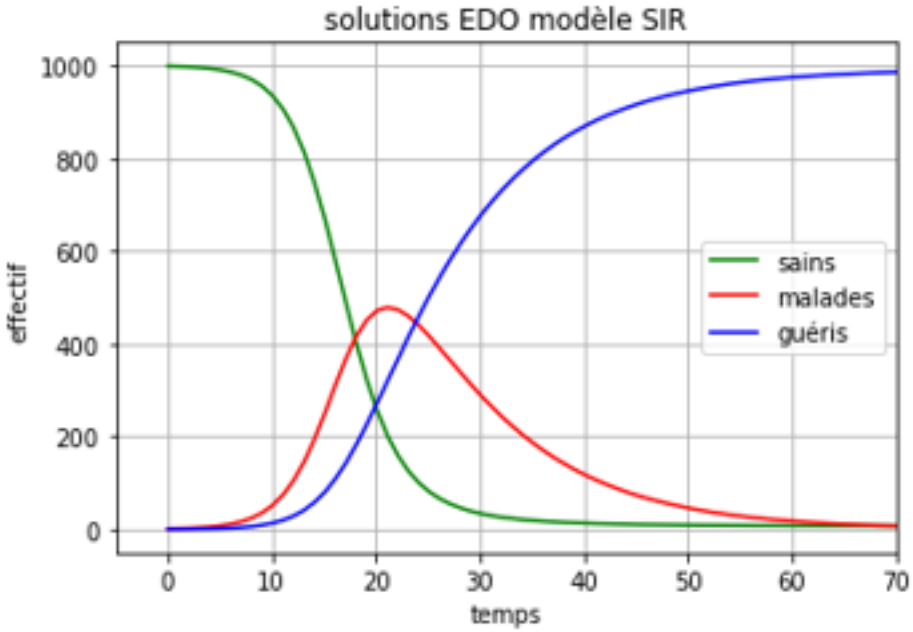
[32]:
```
voir annexe 7
```

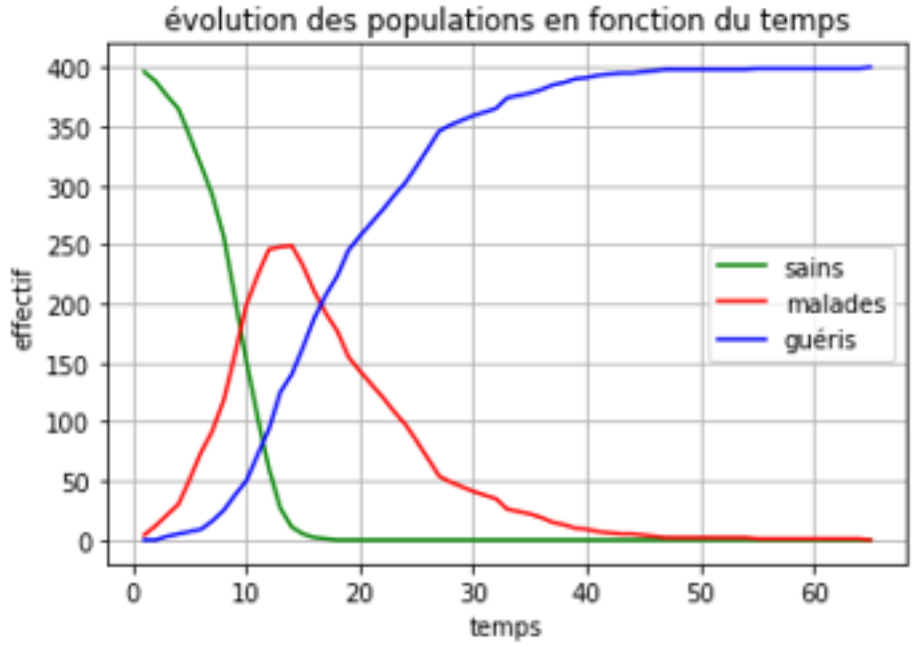


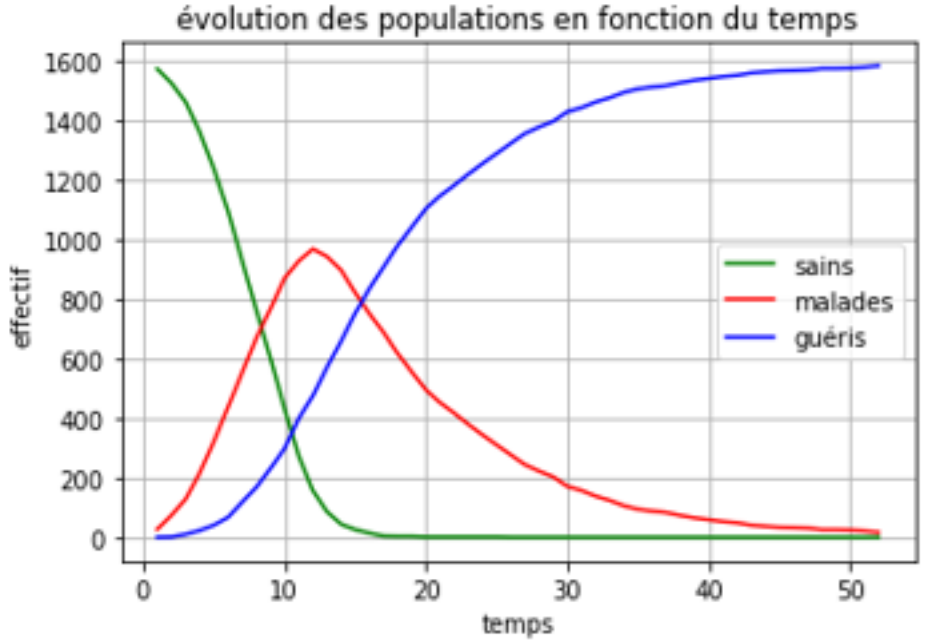


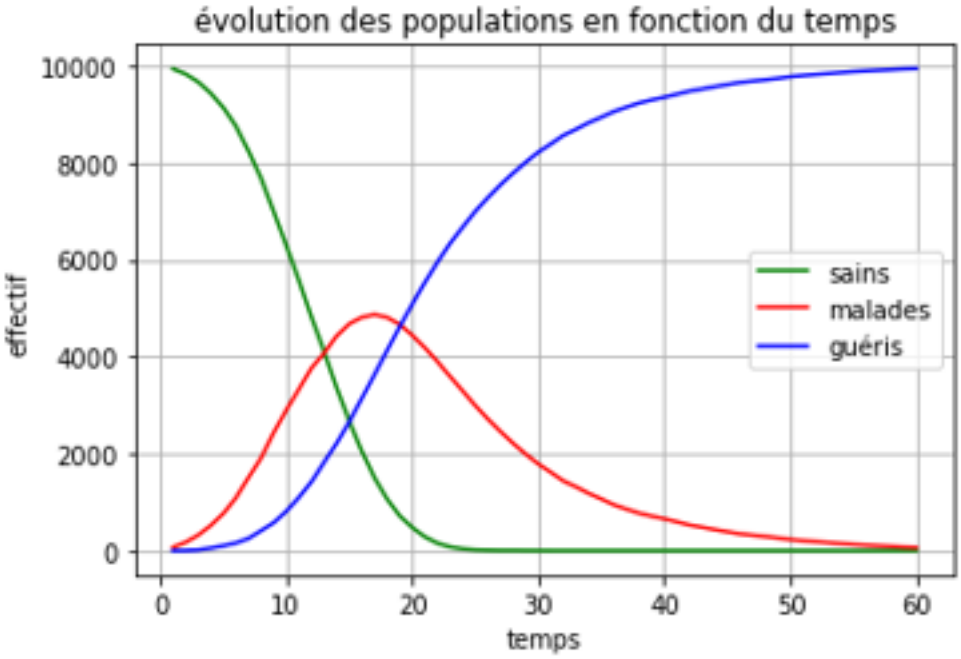
→ Concernant le modèle SIS, on peut remarquer que les graphiques obtenus sont quasiment similaires. Les courbes issues de notre automate sont légèrement oscillantes, contrairement à celles du modèle déterministe, qui sont parfaitement lisses. Cela s'explique par l'aspect probabiliste de notre automate cellulaire, qui fait opposition sur ce point au modèle différentiel, complètement déterministe. Toutefois, plus la population totale de notre automate est importante, moins les courbes obtenues oscillent ; elles tendent à avoir exactement la même allure que les courbes des EDO.

5.3 Comparaison entre notre automate SIR et les solutions du systeme differentiel SIR






évolution des populations en fonction du temps
effectif
temps
sains
malades
guéris
0
200
400
600
800
1000
1200
1400
1600
0
10
20
30
40
50


évolution des populations en fonction du temps
effectif
temps
sains
malades
guéris
0
2000
4000
6000
8000
10000
0
10
20
30
40
50
60

$\rightarrow$ Concernant le modèle SIR, en fonction de la taille de la population de départ, on remarque que les courbes s'éloignent plus ou moins de celles issues du modèle déterministe. Toutefois, même pour un nombre d'individus relativement peu élevé, les allures sont très similaires. Plus on augmente la taille de la population, plus les deux graphes tendent à être identiques et moins les courbes issues de notre modèle oscillent. Ce dernier point est notamment dû au fait que l'augmentation du nombre d'individus « réduit » les effets de l'aspect probabiliste. De plus, pour le modèle SIR, nous avons pu remarquer que notre probabilité p1 d'infection correspond au β du système EDO, tandis que notre probabilité p2 d'infection est égale au 1-γ de ce même modèle. Il était donc légitime en partie 2 de nous servir de la formule du R_0 donnée par le modèle différentiel, même si les résultats obtenus n'ont pas été concluants.

Pour conclure, cette mise en perspective nous a permis de contextualiser notre automate avec la littérature déjà existante. Notre objectif était de modéliser, de la manière la plus précise et à notre niveau, l'évolution d'une épidémie par une approche relativement peu exploitée par la littérature jusqu'ici. Nous avons ainsi pu constater que notre modèle n'est pas absurde ou en complet décalage avec les approches épidémiologiques publiées ; nous retrouvons en effet des points communs essentiels avec l'approche déterministe, qui est de loin la plus répandue et la plus « universelle ».

6 Conclusion

6.1 Les avantages et les limites de notre modèle

Le fait d'avoir choisi de modéliser l'évolution d'une épidémie à l'aide d'un automate cellulaire présente plusieurs avantages importants. Tout d'abord, l'aspect probabiliste-markovien de notre modèle est fidèle à la dimension aléatoire de la propagation d'une épidémie. En effet, dans une population réelle composée d'individus ne présentant pas de facteurs à risques, tout le monde a la même probabilité de tomber malade en fonction de l'état de santé de ses voisins. L'automate cellulaire modélise donc de manière réaliste les phénomènes que l'on peut observer dans une population touchée par une maladie infectieuse. Il est également appréciable de disposer d'une représentation imagée de la propagation d'une maladie en temps réel. De cette manière, on peut pleinement comprendre l'influence de la modification de certains paramètres sur l'évolution de l'épidémie et remarquer l'apparition d'un motif en forme de front progressif. Cet aspect visuel permet donc une meilleure compréhension du processus épidémiologique. En revanche, dans un modèle déterministe tel que celui des EDO, nous n'avons pas la possibilité de visualiser la propagation de l'épidémie à chaque pas de temps. Enfin, notre automate cellulaire est bien adapté à l'étude de petites populations. On pourrait imaginer que ce modèle décrirait assez réalistement l'évolution d'une épidémie dans un petit pays, au sein d'une communauté ou d'une zone géographique faiblement peuplée. Il permet également de suivre individuellement chaque cellule à chaque pas de temps, ce qui peut être appréciable si l'on veut mener une étude très précise.

Hélas, le fait d'avoir choisi une modélisation à l'aide d'un automate cellulaire présente également quelques inconvénients. Le premier – et certainement le plus important – est le fait qu'un automate ne peut modéliser qu'une population finie d'individus. De plus, au-delà d'un certain nombre de cellules, le passage d'un temps T à un temps T+1 commence à demander une quantité non négligeable de calculs, ce qui ralentit également l'algorithme. Lors de notre recherche de la relation entre p1 et p2, nous avons passé un temps considérable à faire tourner l'automate pour réaliser nos différents tests et avoir suffisamment de valeurs pour établir un rapport significatif. Nous travaillions alors sur une grille de 1600 individus ; il aurait été très compliqué – voire impossible – de passer à une population beaucoup plus élevée, à moins d'y passer un temps considérable. De plus, notre modèle, malgré ses qualités, reste tout de même une esquisse simplifiée du comportement d'une épidémie réelle. Notamment, dans un souci de simplicité, nous n'avons pas pris en compte certains paramètres qui ne sauraient être négligés lors d'une véritable étude épidémiologique :

$\rightarrow$ La durée d'incubation de la maladie : dans notre automate, une cellule saine peut devenir malade en une unité de temps, et également guérir en une unité de temps. Dans la réalité et étant donné le contexte actuel, nous savons bien que certaines épidémies mettent plusieurs jours avant de se déclarer complètement et qu'un individu puisse être considéré comme malade. De même, les individus infectés guérissent généralement en plusieurs jours, voire plusieurs semaines, et pas en une seule unité de temps (ie dans notre automate, un jour).

$\rightarrow$ Le nombre de voisins d'un individu peut être très supérieur à 8, ce qui augmente donc considérablement les chances d'une personne saine de devenir infectée au contact d'un voisin malade (car le nombre de voisins croît).

→ Le fait que les individus puissent se déplacer dans toute l'aire géographique étudiée par l'automate : dans notre modèle, tous les individus (ie les cellules) restent fixes et ne peuvent contaminer ou être contaminés uniquement par le contact de leurs voisins, qui sont eux aussi immobiles au cours du temps. Or, dans la réalité, nous savons bien qu'il est extrêmement difficile d'immobiliser complètement une population pendant une grande durée. Pour complexifier le modèle, il faudrait donc prendre en compte les probabilités de mouvement de chaque cellule.

→ Les effets du confinement ou de la vaccination : dans la réalité, des mesures peuvent être prises par les gouvernements pour ralentir la propagation de l'épidémie. Ces restrictions (confinement strict, couvre-feu, vaccination, périmètre limite de déplacement ...) influencent de manière plus ou moins conséquente l'évolution d'une maladie, et sont donc non négligeables lorsque l'on souhaite construire un modèle épidémiologique réaliste.

Toutefois, malgré ces inconvénients, nous avons tout de même réussir à mettre au point un automate cellulaire permettant d'approcher le comportement d'une épidémie. La comparaison avec le modèle déterministe nous pousse à dire que notre automate est cohérent avec les écrits déjà publiés sur ce sujet, et qu'il pourrait être complexifié afin d'améliorer davantage sa représentation de la réalité.

6.2 Auto-critique

Afin de conclure ce rapport, il nous a paru important de dresser un rapide bilan de l'ensemble de notre projet. En premier lieu, nous sommes fiers d'avoir réussi à implémenter nous-même un automate cellulaire cohérent avec les des modèles antérieurs certifiés par la communauté scientifique. Toutefois, la conception et le codage de cet automate furent relativement fastidieux. L'un de nos objectifs premiers étant de pouvoir visualiser graphiquement l'évolution d'une épidémie en temps réel, il a fallu apprendre à utiliser une bibliothèque dont on ne s'était encore jamais servi – à savoir Tkinter – afin de créer une interface dynamique.

Une fois cette étape terminée, nous avons dû sélectionner les règles de fonctionnement de notre automate dans le but de créer un modèle à la fois simple mais un minimum réaliste. Par exemple, nous avons choisi de considérer que chaque cellule était dotée de 8 voisins (et non 4 comme dans certains automates cellulaires). Ensuite, il a fallu déterminer quels paramètres nous paraissaient les plus importants dans un processus de propagation d'une épidémie. Nous avons ainsi choisi de ne garder que trois paramètres d'action :

→ probabilité pour un individu de devenir malade au contact d'au moins un voisin infecté (ie p1)

→ la probabilité pour un individu malade de guérir entre deux instants T et T+1 (ie 1-p2). Nous avons choisi de la noter 1-p2 plutôt que p2 simplement par souci de simplicité de l'écriture du code

→ la taille de la population (ie N)

Nous avons ensuite dû effectuer un très grand nombre de simulations afin de trouver la relation entre p1 et p2; les durées d'exécution s'avérant relativement longues (entre 30 secondes et 4 minutes pour chaque simulation), nous étions anxieux à l'idée de passer autant de temps sur une

approche qui ne nous mènerait peut-être nulle part.

Notre seul véritable regret est le fait de nous être trop longuement attardés sur l'approche markovienne. En effet, nous pensions qu'envisager la grille sous la forme d'une chaîne de Markov nous permettrait d'obtenir de nouveaux résultats importants. Toutefois, nous nous sommes aperçus qu'il était très compliqué (voire impossible) d'étudier des chaînes de Markov sur des matrices de taille supérieure ou égale à 3. Nous regrettons donc d'avoir passé énormément de temps à essayer de coder des matrices de transition complexes pour seulement prouver que le nul de $\mathbb{M}_2(\mathbb{R})$ est un état absorbant. Nous aurions préféré chercher à complexifier les règles de notre automate de départ dans le but de le rendre plus réaliste en tenant compte de nouveaux paramètres (comme une durée d'incubation ou un laps de temps minimal de guérison).

Pour terminer, nous avons retenu beaucoup de points positifs sur ce projet. Ce dernier nous a notamment permis de prendre de nombreuses initiatives et de faire les choses nous-mêmes sans nous inspirer de modèles déjà existants. Nous avons également pu appliquer des concepts vus en cours, comme les chaînes de Markov ou les régressions linéaires, à un contexte totalement nouveau pour nous. Nous sommes également très contents d'avoir choisi un sujet très ouvert ; certes, il ne fallait pas tomber dans le piège de l'éparpillement, mais nous pensons avoir plutôt bien géré cet aspect en étant clairs sur nos objectifs dès le départ. Nous avons apprécié le fait d'être libres d'adopter l'approche qui nous correspondait le mieux, sans être restreints par les limites du sujet.

7 Annexe

7.1 Codes python

On importe ici toutes les bibliothèques qui nous serviront pour la suite.

[22]:
```
import numpy as np
from tkinter import Tk, Canvas, Button, RIGHT, LEFT
from random import uniform
import matplotlib.pyplot as plt
%matplotlib inline
```

annexe 2 :

[23]:
```
# Définition de l'interface graphique
fenetre = Tk()
fenetre.title("personnes saines et infectées")
canvas = Canvas(fenetre, width=a*NbC+1, height=a*NbL+1, highlightthickness=0)
fenetre.wm_attributes("-topmost", True) # afficher la fenetre on top
# Arrêt de l'animation
def stop():
    global flag
    flag=0

# Démarrage de l'animation
def start():
    global flag
    if flag==0:
        flag=1
    iterer()

# Animation pas à pas
def pasapas():
    global flag
    flag=2
    iterer()

# Fonction de traitement du clic gauche de la souris
def  SelectionCellule(event):
    x, y = event.y//a, event.x//a
    tab[x,y] = (tab[x,y]+1)%3
    if tab[x,y]==0:
        color = "white"
    elif tab[x,y]==1:
```

```
        color = "red"
    else:
        color = "blue"
    canvas.itemconfig(cell[x][y], fill=color)


# Appelle la fonction SelectionCellule sur click gauche
canvas.bind("<Button-1>", SelectionCellule)
canvas.pack()

# Définition des boutons de commande
bou1 = Button(fenetre,text='Exit', width=8, command=fenetre.destroy)
bou1.pack(side=RIGHT)
bou2 = Button(fenetre, text='Start', width=8, command=start)
bou2.pack(side=LEFT)
bou3 = Button(fenetre, text='Stop', width=8, command=stop)
bou3.pack(side=LEFT)
bou4 = Button(fenetre, text='Step', width=8, command=pasapas)
bou4.pack(side=LEFT)
```

annexe 3 :

[24]:
```
# Définition de la matrice d'évolution de l'automate
cell = np.zeros((NbL,NbC),dtype=int)

# Initialisation de l'automate
def initialiser_monde():
    tab = np.zeros((NbL,NbC),dtype=int)
    tab[0:NbL,0:NbC] = 0
    # création de la grille
    for x in range(NbL):
        for y in range(NbC):
            cell[x,y] = canvas.create_rectangle((y*a, x*a, (y+1)*a, \
                        (x+1)*a), outline="gray", fill="white")
    return tab

tab=initialiser_monde()
```

annexe 4 :

[25]:
```
# Dessiner toutes les cellules
def dessiner():
    for x in range(NbL):
        for y in range(NbC):
            if tab[x,y]==0:
                coul = "white"
            elif tab[x,y]==1:
                coul = "red"
            else:
                coul = "blue"
            canvas.itemconfig(cell[x][y], fill=coul)
```

annexe 5 :

[26]:
```
#visualiser le nombre de malades
def compte():
    global tab,malades_vivants
    nb_sains=0
    nb_infectés=0
    nb_guéris=0
    for i in range(NbL):
        for j in range(NbC):
            if tab[i,j]==0:
                nb_sains+=1
            elif tab[i,j]==1:
                nb_infectés+=1
            else:
                nb_guéris+=1
    malades_vivants=(nb_infectés!=0)
    return nb_sains,nb_infectés,nb_guéris

#on enregistre les données de l'automate à chaque passage pour pouvoir tracer le␣
 ↪graphe
def ajouter_graphique():
    global nb_passages,L,temps
    nb_passages+=1
    nb_sains,nb_infectés,nb_guéris=compte()
    L.append([nb_sains,nb_infectés,nb_guéris])
    temps.append(nb_passages)
```

annexe 6 :

[31]:
```
# lancement de l'automate
flag = 0
initialiser_malades()
dessiner()
fenetre.mainloop()
```

annexe 7 :

[ ]:
```
fig=plt.figure(1)
plt.title("évolution des populations en fonction du temps")
plt.xlabel('nombre de jours')
plt.ylabel('effectifs')
plt.plot(temps,np.array(L)[:,0],'-g',temps,np.array(L)[:,1],'-r',temps,np.
 ↪array(L)[:,2],'-b')
plt.legend(['sains','malades', 'guéris'],loc='best')
plt.grid(True)
plt.show(block=False)
```

annexe 8 :

[ ]:
```
x_sum = 0
y_sum=0

for xi in liste_p1_frontiere:
    x_sum += xi

for yi in liste_p2_frontiere:
    y_sum += yi

x2_sum = 0.
for xi in liste_p1_frontiere:
    x2_sum += xi**2

xy_sum = 0.
for xi, yi in zip(liste_p2_frontiere,liste_p1_frontiere):
    xy_sum += xi * yi

npoints = len(liste_p1_frontiere)
a = (npoints * xy_sum - x_sum * y_sum) / (npoints * x2_sum - x_sum**2)
b = (x2_sum * y_sum - x_sum * xy_sum) / (npoints * x2_sum - x_sum**2)

def regLin(x, y):
```

```
    x = np.array(x)
    y = np.array(y)
    npoints = len(x)
    a = (npoints * (x*y).sum() - x.sum()*y.sum()) / (npoints*(x**2).sum() - (x.
 ↪sum())**2)
    b = ((x**2).sum()*y.sum() - x.sum() * (x*y).sum()) / (npoints * (x**2).sum()␣
 ↪- (x.sum())**2)
    return a, b

a, b = regLin(liste_p2_frontiere,liste_p1_frontiere)
y=[]
for xi in liste_p2_frontiere:
    y.append(xi*a+b)
plt.scatter(liste_p2_frontiere,liste_p1_frontiere)
plt.plot(liste_p2_frontiere,y)
plt.xlabel("p2")
plt.ylabel("p1")
plt.title("Evolution de p1 en fonction de p2")
plt.grid()
plt.show()
print("coeff directeur = ",a)
print("ordonnée à l'origine = ",b)
```

annexe 9 :

[19]:
```
X1=[[0,0],[0,0]]
X2=[[1,0],[0,0]]
X3=[[0,1],[0,0]]
X4=[[0,0],[1,0]]
X5=[[0,0],[0,1]]
X6=[[1,1],[0,0]]
X7=[[1,0],[1,0]]
X8=[[1,0],[0,1]]
X9=[[0,1],[1,0]]
X10=[[0,1],[0,1]]
X11=[[0,0],[1,1]]
X12=[[1,1],[1,0]]
X13=[[1,1],[0,1]]
X14=[[1,0],[1,1]]
X15=[[0,1],[1,1]]
X16=[[1,1],[1,1]]
liste_X_possibles=[X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12,X13,X14,X15,X16]
```

annexe 10 :

[20]:
```
def transfo_mu(i):
    if i:
        return np.array([0,1],dtype=float)
    else:
        return np.array([1,0],dtype=float)

def transfo_X(X):
    res=[]
    for i in range(NbL2):
        temp=[]
        for j in range(NbC2):
            temp.append(transfo_mu(X[i,j]))
        res.append(temp)
    return np.array(res)
```

annexe 11 :

[21]:
```
def afficher_proba_totale_changement(Xinit,Xfinal):
    X=matrice_proba_changement(Xinit,Xfinal,M)
    res=1
    for i in range(NbL2):
        for j in range(NbC2):
            res*=X[i,j]
    return res
```

annexe 12 :

[22]:
```
def somme_ligne(M):
    S=np.zeros(16)
    for i in range(16):
        for j in range(16):
            S[i]+=M[i,j]
    return S
```

annexe 13 :

[23]:
```
def P_puissance_n(n,P):
    P_n=P
    for k in range(n):
        P_n=np.dot(P_n,P)
    return P_n
```

annexe 14 :

[ ]:
```
P=matrice_transition_globale(liste_X_possibles)
n=100
res1,liste_n=proba_arriver_au_nul(n,P)
plt.plot(liste_n,res1[:,0], c='r')
plt.plot(liste_n,res1[:,1], c='b')
plt.plot(liste_n,res1[:,5], c='g')
plt.plot(liste_n,res1[:,12], c='y')
plt.plot(liste_n,res1[:,15], c='brown')
plt.ylabel("proba d'être à l'élement absorbant")
plt.xlabel("nb de jours")
plt.grid()
plt.show()
```

7.2 Bibliographie

A cellular automaton model for the effects of population movement and vaccination on epidemic propagation G. Ch. Sirakoulis, I. Karafyllidis*, A. Thanailakis

On the basic reproduction number and the topological properties of the contact network : An epidemiological study in mainly locally connected cellular automata P.H.T. Schimit a, L.H.A. Monteiro b,a,

→ http ://www.tangentex.com/ACEpidemie.htmPar2

→ https ://tel.archives-ouvertes.fr/tel-01276668/document

→ https ://www.researchgate.net/profile/George-Milne-3/publication /228966629

Epidemic-Modelling-Using-Cellular-Automata/links/0f317534deef0bbfd5000000

/Epidemic-Modelling-Using-Cellular-Automata.pdf

→ https ://www.ncbi.nlm.nih.gov/pmc/articles/PMC7120465/pdf /978-3-540-33995-3/Chapter/20.pdf