TP : Propagation d'une épidémie

1 Introduction

Ce TP porte sur l'étude numérique de la propagation d'une épidémie dans une population. Nous aborderons se problème en utilisant une modélisation à base d'un *automate cellulaire*, c'est-à-dire que l'on va modéliser les individus d'une population par les cases d'une grille, chaque case étant dans un état donné. On détermine l'évolution de l'épidémie en calculant un nouvel état de la grille en fonction de l'état précédent.

Nous allons essayer de déterminer l'évolution du nombre de personnes infectées au cours du temps et voir à quelle condition l'épidémie peut devenir une pandémie.

2 Principe de la méthode

Nous allons modéliser une population par une grille carrée de $n \times n$ cases. Chaque case peut être dans trois états différents :

- État 0 : `SAIN`
- État 1 : `MALADE`
- État 2 : `IMMUNISE`

On commence avec une grille dans laquelle tous les individus sont sont dans l'état `SAIN` sauf l'individu situé au centre de la grille qui est dans l'état `MALADE`. Ensuite on fait évoluer la grille en suivant les règles suivantes :

- une case dans un état `IMMUNISE` à l'étape n reste dans le même état à l'étape $n+1$;
- une case dans un état `MALADE` à l'étape n passe à l'état `IMMUNISE` à l'étape $n+1$;
- une case dans un état `SAIN` à l'étape n reste dans le même état à l'étape $n+1$ **sauf** si l'une des cases voisines est dans l'état `MALADE`, auquel cas, elle passe à l'état `MALADE` avec une probabilité p_c et à l'état `IMMUNISE` avec une probabilité $1-p_c$.

p_c est la probabilité qu'à un individu de devenir malade lorsqu'au moins un de ses voisins est malade. S'il ne devient pas malade, c'est qu'il est immunisé. On considère que chaque case de la grille possède 4 voisins qui sont les cases avec lesquelles elle partage un côté (les cases en diagonale ne comptent pas). On ignore le cas des cases qui sont au bord de la grille, leur état ne sera pas modifié au cours de la simulation.

On calcule les états successifs de la grille jusqu'à ce que l'une des conditions suivantes soit remplie :

- il n'y a plus d'individu malade sur la grille, l'épidémie s'est donc arrêtée ;
- l'épidémie a atteint un individu du bord de la grille, dans ce cas on considère que l'épidémie est devenue une pandémie (épidémie qui se répand sans limite). Dans le cas présent, comme les cases du bord de la grille ne changent jamais d'état, on considérera que l'on a une pandémie si une case voisine d'une case de bord de grille est atteinte.

3 Programmation

La grille sera représentée par une liste `G` de n listes d'entiers, l'entier `G[i][j]` représente l'état de l'individu se situant dans la case de coordonnées (i, j). On pourra représenter les trois états considérés de la manière suivante :

```
SAIN = 0
MALADE = 1
IMMUNISE = 2
```

1. Écrire une fonction d'entête :

   ```
   def initGrille(n:int) -> list :
   ```

   qui initialise une grille de $n \times n$ individus tous dans l'état `SAIN`, sauf l'individu situé au centre de la grille qui est dans l'état `MALADE`.

2. Écrire une fonction d'entête :

```
def stats(G:list) -> list :
```

qui renvoie une liste de trois entiers `[sains, malades, immunises]` correspondant respectivement au nombre d'individus dans l'état `SAIN`, `MALADE`, `IMMUNISE`.

3. On définit la fonction suivante :

```
def tirage(p:float) -> bool :
  if random.random()<p:
    return True
  else:
    return False
```

Indiquer en quelques mots ce que fait cette fonction et en quoi elle peut être utile dans le problème qui nous intéresse.

4. Écrire une fonction d'entête :

```
def majGrille(G:list, pc:float) -> list :
```

qui prend en paramètre une grille `G` et une probabilité de contamination `pc` et qui renvoie la grille après lui avoir fait subir une étape de la simulation.

5. Écrire une fonction d'entête :

```
def estPandemie(G:list) -> bool :
```

qui prend en paramètre une grille `G` et qui renvoie `True` si l'épidémie a atteint le bord de la grille et `False` sinon.

6. Écrire une fonction d'entête :

```
def simulation(n:int, pc:float) -> [bool, list] :
```

qui simule la propagation de l'épidémie sur une grille de côté `n`, avec une probabilité de contamination `pc`. La simulation continue jusqu'à ce que l'épidémie se soit arrêtée (plus de malade sur la grille) ou qu'elle ait atteint le stade de pandémie (un malade dans une case voisine du bord de grille). Si l'épidémie s'est arrêtée, le premier élément de la liste renvoyée par la fonction est `False`, cet élément vaut `True` si c'est une pandémie. Le second élément renvoyé par la fonction est une liste contenant toutes les grilles successives calculées au cours de la simulation.

En jouant un peu avec la probabilité de contamination, sur une grille suffisamment grande ($n = 100$), on se rend compte que pour des valeurs faibles de p_c ($p_c < 0{,}3$), l'épidémie s'arrête toujours. Pour des valeurs grandes de p_c ($p_c > 0{,}7$) l'épidémie devient systématiquement une pandémie. Pour des valeurs intermédiaires de la probabilité de contamination, l'épidémie évolue parfois en pandémie, et parfois elle s'arrête.

Pour visualiser la propagation de l'épidémie, on pourra utiliser le code suivant :

```
def animeSimul(n, pc):
    pandemie, mem = simulation(n,pc)
    fig = plt.figure()
    norm = matplotlib.colors.Normalize(vmax=2, vmin=0)
    image = plt.imshow(mem[0], norm=norm)
    def animate(i):
        image.set_data(mem[i])
        return (image,)
    ani = animation.FuncAnimation(fig, animate, frames=len(mem), repeat=False, interval=20)
    plt.show()
```

On prendra soin d'importer les bibliothèques nécessaires en plaçant les lignes suivantes au début du programme :

```
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import matplotlib
```

On voudrait déterminer en fonction de p_c, la probabilité qu'a l'épidémie de devenir une pandémie. Pour cela, on effectue un grand nombre de simulations (environ 100) et on calcule la proportion de ces simulations qui ont évolué en pandémie.

7. Écrire une fonction d'entête :

```
def probaPandemie(n:int, pc:float) -> float :
```

qui effectue 100 simulations sur une grille de côté `n` avec une probabilité de contamination `pc` et qui renvoie la probabilité que l'épidémie se transforme en pandémie.

Enfin, ou cherche à déterminer la valeur limite p_l de la probabilité de contamination pour que l'épidémie ait une probabilité de devenir une pandémie supérieur à 0,5. Pour que le calcul soit le plus efficace possible, on utilisera une méthode de dichotomie qui devrait permettre de déterminer p_l à 10^{-2} près.

8. Écrire une fonction d'entête :

```
def trouveProbaLimite(n:int) -> float :
```

qui détermine la probabilité limite p_l à 10^{-2} près en utilisant une grille de côté `n` par une méthode de dichotomie.