Épreuve d'informatique – Propagation d'une épidémie

Certaines questions demandent d'écrire du code python, vous veillerez autant que possible à utiliser une syntaxe valide et à indenter correctement le code (utilisez des lignes verticales pour marquer les différents niveaux d'indentation)

L'étude de la propagation des épidémies joue un rôle important dans les politiques de santé publique. Les modèles mathématiques ont permis de comprendre pourquoi il a été possible d'éradiquer la variole à la fin des années 1970 et pourquoi il est plus difficile d'éradiquer l'apparition d'épidémies de grippe tous les hivers. Aujourd'hui, des modèles de plus en plus complexes et puissants sont développés pour prédire la propagation d'épidémies à l'échelle planétaire telles que le SRAS, le virus H5N1 ou le virus Ebola. Ces prédictions sont utilisées par les organisations internationales pour établir des stratégies de prévention et d'intervention.

Le travail sur ces modèles mathématiques s'articule autour de trois thèmes principaux : traitement de base des données, simulation numérique (par plusieurs types de méthodes), identification des paramètres intervenant dans les modèles à partir de données expérimentales. Ces trois thèmes sont abordés dans le sujet. *Les parties sont indépendantes.*

Dans tout le problème, on peut utiliser une fonction traitée précédemment. On suppose que les bibliothèques `numpy` et `random` ont été importées par :

```
import numpy as np
import random as rd
```

Partie I. Tri

Dans le but ultérieur de réaliser des études statistiques, on souhaite se doter de quelques fonctions agissant sur une liste L de flottants.

1. Écrire une fonction `plusGrand(L)` qui prend en argument une liste de flottants et qui renvoie la plus grande valeur de la liste (l'utilisation de la fonction `max` n'est pas autorisée). Quelle est la complexité de cette fonction en fonction du nombre n d'éléments de `L` ?
2. Écrire une fonction `moyenne(L)` qui prend en argument une liste de flottants et qui renvoie la moyenne de ses éléments. Quelle est la complexité de cette fonction en fonction du nombre n d'éléments de `L` ?

On se donne également la fonction `tri` suivante, écrite en Python :

```
def tri(L):
    n=len(L)
    for i in range(1,n):
        j=i
        x=L[i]
        while 0<j and x<L[j-1]:
            L[j]=L[j-1]
            j=j-1
        L[j]=x
```

3. Lors de l'appel `tri(L)` lorsque L est la liste `[5,2,3,1,4]`, donner le contenu de la liste L à la fin de chaque itération de la boucle `for`.
4. Soit L une liste non vide d'entiers ou de flottants. Montrer que « la liste `L[0:i+1]` (avec la convention Python) est triée par ordre croissant à l'issue de l'itération i »est un invariant de boucle. En déduire que `tri(L)` trie la liste L.
5. Quel est le nom de l'algorithme de tri utilisé dans la fonction `tri(L)` ?
6. Évaluer la complexité dans le meilleur des cas et dans le pire des cas de l'appel `tri(L)` en fonction du nombre d'éléments de L.

On souhaite, partant d'une liste constituée de couples (chaîne, entier), trier la liste par ordre croissant de l'entier associé suivant le fonctionnement suivant :

```
>>> L=[['Bresil',76],['Kenya',26017],['Ouganda',8431]]
>>> tri_chaine(L)
>>> L
[['Bresil',76],['Ouganda',8431],['Kenya',26017]]
```

7. Modifier la fonction `tri` précédente pour créer une fonction `tri_chaine` qui réalise cette opération.

Pour suivre la propagation des épidémies, de nombreuses données sont recueillies par les institutions internationales comme l'OMS. Par exemple, pour le paludisme, on dispose de deux tables :

— La table `palu` recense le nombre de nouveaux cas confirmés et le nombre de décès liés au paludisme ; certaines lignes de cette table sont données en exemple (on précise que `iso` est un identifiant unique pour chaque pays) :

nom	iso	annee	cas	deces
Bresil	BR	2009	309 316	85
Bresil	BR	2010	334 667	76
Kenya	KE	2010	898 531	26 017
Mali	ML	2011	307 035	2 128
Ouganda	UG	2010	1 581 160	8 431
...				

— la table `demographie` recense la population totale de chaque pays ; certaines lignes de cette table sont données en exemple :

pays	periode	pop
BR	2009	193 020 000
BR	2010	194 946 000
KE	2010	40 909 000
ML	2011	14 417 000
UG	2010	33 987 000
...		

8. Au vu des données présentées dans la table `palu`, parmi les attributs `nom`, `iso` et `annee`, quels attributs peuvent servir de clé primaire ? Un couple d'attributs pourrait-il servir de clé primaire ? (on considère qu'une clé primaire peut posséder plusieurs attributs). Si oui, en préciser un.
9. Écrire une requête en langage SQL qui récupère depuis la table `palu` toutes les données de l'année 2010 qui correspondent à des pays où le nombre de décès dus au paludisme est supérieur ou égal à 1 000.

On appelle *taux d'incidence d'une épidémie* le rapport du nombre de nouveaux cas pendant une période donnée sur la taille de la population-cible pendant la même période. Il s'exprimer généralement en « nombre de nouveaux cas pour 100 000 personnes par année ». Il s'agit d'un des critères les plus importants pour évaluer la fréquence et la vitesse d'apparition d'une épidémie.

Par exemple avec les données des tables ci-dessus, on peut calculer qu'en 2009, le taux d'incidence du paludisme au Brésil était de :

$$\eta = \frac{309\,316}{193\,020\,000} = 1{,}6 \times 10^{-3} = 160\ \text{cas}/100000\ \text{personnes}$$

10. Écrire une requête en langage SQL qui détermine le taux d'incidence du paludisme en 2011 pour les différents pays de la table `palu`.
11. Écrire une requête en langage SQL permettant de déterminer le nom du pays ayant eu le plus grand nombre de nouveaux cas de paludisme en 2010 (on pourra supposer qu'il n'y a pas de pays *ex æquo* pour les nombres de cas).

On considère la requête SQL suivante :

```sql
SELECT nom, deces FROM palu WHERE annee=2010;
```

On suppose que le résultat de cette requête a été converti en une liste Python stockée dans la variable `deces2010` et constituée de couples (chaîne,entier).

12. À l'aide des questions précédentes, écrire une instruction Python pour trier la liste `deces2010` par ordre croissant du nombre de décès dus au paludisme en 2010 ?

Partie II. Modèle à compartiments

On s'intéresse ici à une première méthode de simulation numérique.

Les modèles compartimentaux sont des modèles déterministes où la population est divisée en plusieurs catégories selon leurs caractéristiques et leur état par rapport à la maladie. On considère dans cette partie un modèle à quatre compartiments disjoints : sains (S), infectés (I), rétablis(R, ils sont immunisés) et décédés (D).

Le changement d'état des individus est gouverné par un système d'équations différentielles obtenues en supposant que le nombre d'individus nouvellement infectés (c'est-à-dire le nombre de ceux qui quittent le compartiment S) pendant un intervalle de temps donné est proportionnel au produit du nombre d'individus infectés avec le nombre d'individus sains.

En notant $S(t)$, $I(t)$, $R(t)$ et $D(t)$ la fraction de la population appartenant à chacune des quatre catégories à l'instant t, on obtient le système :

$$\begin{cases} \dfrac{d}{dt}S(t) = -rS(t)I(t) \\ \dfrac{d}{dt}I(t) = rS(t)I(t) - (a+b)I(t) \\ \dfrac{d}{dt}R(t) = aI(t) \\ \dfrac{d}{dt}D(t) = bI(t) \end{cases} \tag{1}$$

avec r le taux de contagion, a le taux de guérison et b le taux de mortalité. On suppose qu'à l'instant initial $t = 0$, on a $S(0) = 0{,}95$, $I(0) = 0{,}05$ et $R(0) = D(0) = 0$.

13. Préciser un vecteur X et une fonction f (en donnant son domaine de définition et son expression) tels que le système différentiel (1) s'écrive sous la forme
$$\frac{d}{dt}X = f(X).$$

14. Compléter la ligne 4 du code suivant (on précise que `np.array` permet de créer un tableau `numpy` à partir d'une liste donnant ainsi la possibilité d'utiliser les opérateurs algébriques). La variable `X` est un vecteur `numpy` représentant le vecteur X défini ci-dessus. La fonction `f` renvoie le vecteur dérivé de `X`.

```
1  def f(X):
2      """Fonction definissant l'equation differentielle"""
3      global r,a,b
4      # a completer
5
6  # Parametres
7  tmax=25.
8  r=1.
9  a=0.4
10 b=0.1
11 X0=np.array([0.95,0.05,0.,0.])
12
13 N=250
14 dt=tmax/N
15
16 t=0
17 X=X0
18 tt=[t]
19 XX=[X]
20
21 # Methode d'Euler
22 for i in range(N):
23     t=t+dt
24     X=X+dt*f(X)
25     tt.append(t)
26     XX.append(X)
```

15. La figure 1 représente les quatre catégories en fonction du temps obtenues en effectuant deux simulations : la première avec $N = 7$ correspondant aux points (cercle, carré, losange, triangle) et la seconde avec $N = 250$ correspond aux courbes. Expliquer la différence entre ces deux simulations. Quelle simulation a nécessité le temps de calcul le plus long ?

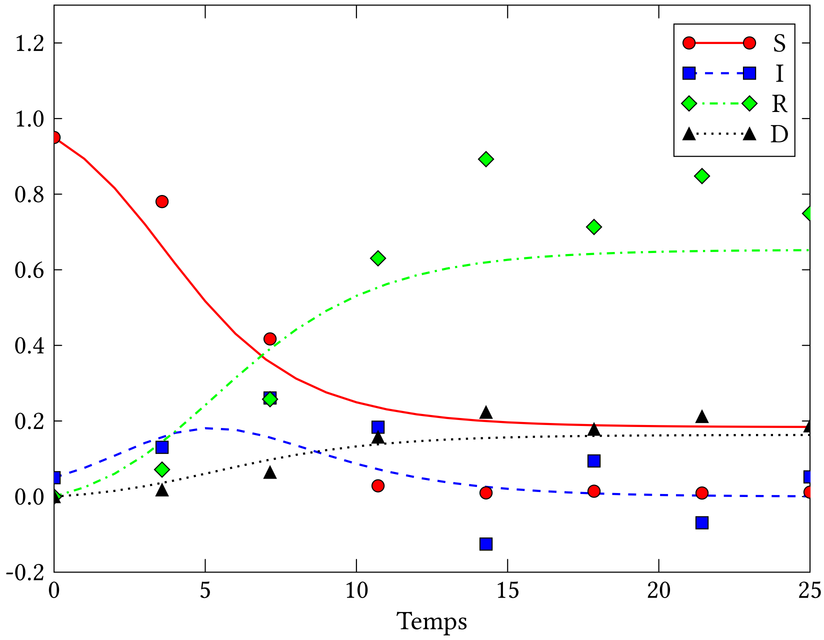

FIGURE 1 – Représentation graphique des quatre catégories S, I, R et D en fonction du temps pour $N = 7$ (points) et $N = 250$ (courbes)

En pratique, de nombreuses maladies possèdent une phase d'incubation pendant laquelle l'individu est porteur de la maladie mais ne possède par de symptômes et n'est pas contagieux. On peut prendre en compte cette phase d'incubation à l'aide du système à retard suivant :

$$\begin{cases} \frac{d}{dt}S(t) = -rS(t)I(t-\tau) \\ \frac{d}{dt}I(t) = rS(t)I(t-\tau) - (a+b)I(t) \\ \frac{d}{dt}R(t) = aI(t) \\ \frac{d}{dt}D(t) = bI(t) \end{cases} \tag{2}$$

où τ est le temps d'incubation. On suppose alors que pour tout $t \in [-\tau, 0]$, $S(t) = 0,95$, $I(t) = 0,05$ et $R(t) = D(t) = 0$.

En notant `tmax` la durée des mesures et `N` un entier donnant le nombre de pas, on définit le pas de temps `dt=tmax/N`. On suppose que $\tau = p \times dt$ où p est un entier ; ainsi p est le nombre de pas de retard.

Pour résoudre numériquement ce système d'équations différentielles à retard (avec `tmax=25, N=250` et `p=50`), on a écrit le code suivant :

```
def f(X,Itau):
    """
    Fonction definissant l'equation differentielle
    Itau est le valeur de I(t-p*dt)
    """
    global r,a,b
    # a completer

# Parametres
r=1.
a=0.4
b=0.1
X0=np.array([0.95,0.05,0.,0.])

tmax=25.
N=250
dt=tmax/N
p=50

t=0
```

```
21 X=X0
22 tt=[t]
23 XX=[X]
24
25 # Methode d'Euler
26 for i in range(N):
27     t=t+dt
28     # a completer
29     tt.append(t)
30     XX.append(X)
```

16. Compléter les lignes 7 et 28 du code précédent (utiliser autant de lignes que nécessaire).

Partie III. Modélisation dans des grilles

On s'intéresse ici à une seconde méthode de simulation numérique (dite *par automates cellulaires*).

Dans ce qui suit, on appelle grille de taille $n \times n$ une liste de n listes de longueur n, où n est un entier strictement positif.

Pour mieux prendre en compte la dépendance spatiale de la contagion, il est possible de simuler la propagation d'une épidémie à l'aide d'une grille. Chaque case de la grille peut être dans un des quatre états suivants : saine, infectée, rétablie, décédée. On choisit de représenter ces quatre états par les entiers :

0 (Sain), 1 (Infecté), 2 (Rétabli) et 3 (Décédé)

L'état des cases d'une grille évolue au cours du temps selon des règles simples. On considère un modèle où l'état d'une case à l'instant $t+1$ ne dépend que de son état à l'instant t et de l'état de ses huit cases voisines à l'instant t (une case du bord n'a que 5 cases voisines et trois pour une case d'un coin). Les *règles de transition* sont les suivantes :

— une case décédée reste décédée ;
— une case infectée devient décédée avec une probabilité p_1 ou rétablie avec une probabilité $(1-p_1)$;
— une case rétablie reste rétablie ;
— une case saine devient infectée avec une probabilité p_2 si elle a au moins une case voisine infectée et reste saine sinon.

On initialise toutes les cases dans l'état sain, sauf une case choisie au hasard dans l'état infecté.

17. On a écrit en Python la fonction `grille(n)` suivante :

```
def grille(n):
    M=[]
    for i in range(n):
        L=[]
        for j in range(n): L.append(0)
        M.append(L)
    return M
```

Décrire ce que retourne cette fonction.

On pourra dans la question suivante utiliser la fonction `randrange(p)` de la bibliothèque `random` qui, pour un entier positif p, renvoie un entier choisi aléatoirement entre 0 et $p-1$ inclus.

18. Écrire en Python une fonction `init(n)` qui construit une grille G de taille $n \times n$ ne contenant que des cases saines, choisit aléatoirement une des cases et la transforme en case infectée, et enfin renvoie G.
19. Écrire en Python une fonction `compte(G)` qui a pour argument une grille G et renvoie la liste `[n0,n1,n2,n3]` formée des nombres de cases dans chacun des quatre états.

D'après les règles de transition, pour savoir si une case saine peut devenir infectée à l'instant suivant, il faut déterminer si elle est exposée à la maladie, c'est-à-dire si elle possède au moins une case infectée dans son voisinage. Pour cela, on écrit en Python la fonction `est_exposee(G,i,j)` suivante :

```
def est_exposee(G,i,j):
    n=len(G)
    if i == 0 and  j == 0:
        return (G[0][1]-1)*(G[1][1]-1)*(G[1][0]-1) == 0
    elif i == 0 and j == n-1:
        return (G[0][n-2]-1)*(G[1][n-2]-1)*(G[1][n-1]-1) == 0
```

```
7      elif i == n-1 and j == 0:
8          return (G[n-1][1]-1)*(G[n-2][1]-1)*(G[n-2][0]-1) == 0
9      elif i == n-1 and j == n-1:
10         return (G[n-1][n-2]-1)*(G[n-2][n-2]-1)*(G[n-2][n-1]-1) == 0
11     elif  i == 0:
12         # a completer
13     elif i == n-1:
14         return (G[n-1][j-1]-1)*(G[n-2][j-1]-1)*(G[n-2][j]-1)*
15                (G[n-2][j+1]-1)*(G[n-1][j+1]-1) == 0
16     elif j == 0:
17         return (G[i-1][0]-1)*(G[i-1][1]-1)*(G[i][1]-1)*
18                (G[i+1][1]-1)*(G[i+1][0]-1) == 0
19     elif j == n-1:
20         return (G[i-1][n-1]-1)*(G[i-1][n-2]-1)*(G[i][n-2]-1)*
21                (G[i+1][n-2]-1)*(G[i+1][n]-1) == 0
22     else:
23         # a completer
```

20. Quel est le type du résultat renvoyé par la fonction `est_exposee` ?
21. Compléter les lignes 12 et 23 de la fonction `est_exposee`.
22. Écrire une fonction `suivant(G,p1,p2)` qui fait évoluer toutes les cases de la grille G à l'aide des règles de transition et renvoie une nouvelle grille correspondant à l'instant suivant. Les arguments `p1` et `p2` sont les probabilités qui interviennent dans les règles de transition pour les cases infectées et les cases saines. On pourra utiliser la fonction `bernoulli(p)` suivante qui simule une variable aléatoire de Bernoulli de paramètre p : `bernoulli(p)` vaut 1 avec la probabilité p et 0 avec la probabilité $1-p$.

```
def bernoulli(p):
    x=rd.random()
    if x <=p:
        return 1
    else:
        return 0
```

On reproduit ci-dessous le descriptif de la documentation Python concernant la fonction `random` de la bibliothèque `random` :

```
random.random()
    Return the next random floating point number in the range [0.0, 1.0]
```

Avec les règles de transition du modèle utilisé, l'état de la grille évolue entre les instants t et $t+1$ tant qu'il existe au moins une case infectée.

23. Quelle est la valeur de la proportion des cases infectées `x1` à la fin d'une simulation ?
24. Écrire en Python une fonction `simulation(n,p1,p2)` qui réalise une simulation complète avec une grille de taille $n \times n$ pour les probabilités `p1` et `p2` et renvoie la liste `[x0,x1,x2,x3]` formée des proportions de cases dans chacun des quatre états à la fin de la simulation (une simulation s'arrête lorsque la grille n'évolue plus).
25. Quelle relation vérifient `x0,x1,x2` et `x3` ? Comment obtenir à l'aide des valeurs de `x0,x1,x2` et `x3` la valeur `x_atteinte` de la proportion des cases qui ont été atteintes par la maladie pendant une simulation ?

On fixe `p1` à 0,5 et on calcule la moyenne des résultats de plusieurs simulations pour différentes valeurs de `p2`. On obtient la courbe de la figure 2.

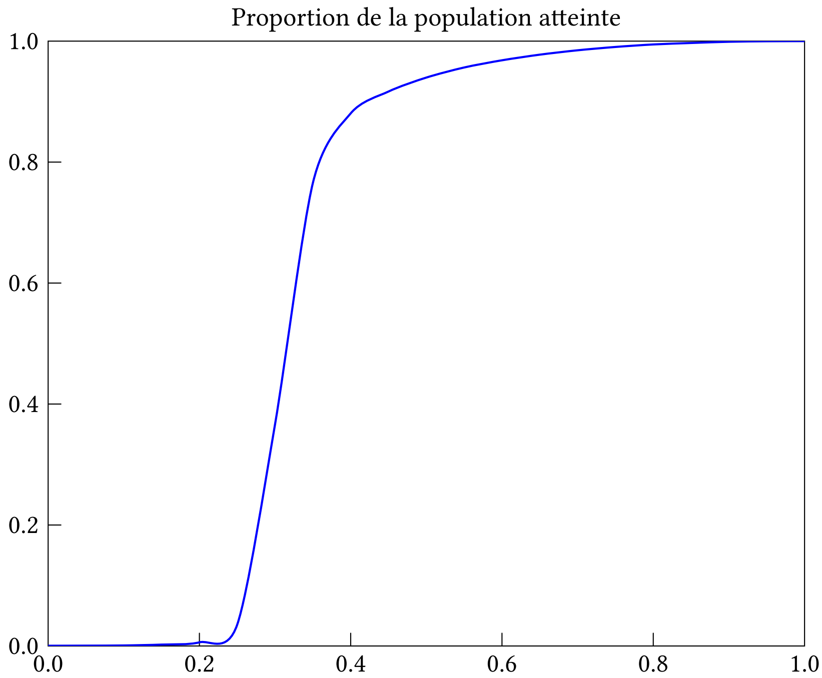

FIGURE 2 – Représentation de la proportion de la population qui a été atteinte par la maladie pendant la simulation en fonction de la probabilité $p2$

26. On appelle *seuil critique de pandémie* la valeur de p2 à partir de laquelle plus de la moitié de la population a été atteinte par la maladie à la fin de la simulation. On suppose que les valeurs de p2 et `x_atteinte` utilisées pour tracer la courbe de la figure 2 on été stockées dans deux listes de même longueur `Lp2` et `Lxa`.
    Écrire en Python une fonction `seuil(Lp2,Lxa)` qui détermine par dichotomie un encadrement `[p2cmin,p2cmax]` du seuil critique de pandémie avec la plus grande précision possible. On supposera que la liste `Lp2` croît de 0 à 1 et que la liste `Lxa` des valeurs correspondantes est croissante.

Pour étudier l'effet d'une campagne de vaccination, on immunise au hasard à l'instant initial une fraction q de la population. On a écrit la fonction `init_vac(n,q)`.

```
1  def init_vac(n,q):
2      G = init(n)
3      nvac = int(q*n**2)
4      k = 0
5      while k < nvac:
6          i = rd.randrange(n)
7          j = rd.randrange(n)
8          if G[i][j] == 0:
9              G[i][j] = 2
10             k += 1
11     return G
```

27. Peut-on supprimer le test en ligne 8 ?
28. Que renvoie l'appel `init_vac(5,0.2)` ?