TP 09

Tableau, complexité et tracé de fonctions

Sources : exercice 1 : A. Troesch, J.-P. Becirspahic

Savoirs et compétences :

- ❒ *AA.C9 : Choisir un type de données en fonction d'un problème à résoudre*
- ❒ *AA.S11 : Manipulation de quelques structures de données.*

Activité 1 : Modélisation de la percolation

La percolation [1] désigne le passage d'un fluide à travers un solide poreux. Ce terme fait bien entendu référence au café produit par le passage de l'eau à travers une poudre de café comprimée, mais dans un sens plus large peut aussi bien s'appliquer à l'infiltration des eaux de pluie jusqu'aux nappes phréatiques ou encore à la propagation des feux de forêt par contact entre les feuillages des arbres voisins.

L'étude scientifique des modèles de percolation s'est développée à partir du milieu du XXe siècle et touche aujourd'hui de nombreuses disciplines, allant des mathématiques à l'économie en passant par la physique et la géologie.

1.1 Choix d'un modèle

Nous allons aborder certains phénomènes propres à la percolation par l'intermédiaire d'un modèle très simple : une grille carrée $n \times n$, chaque case pouvant être ouverte (avec une probabilité p) ou fermée (avec une probabilité $1-p$). La question à laquelle nous allons essayer de répondre est la suivante : est-il possible de joindre le haut et le bas de la grille par une succession de cases ouvertes adjacentes ?

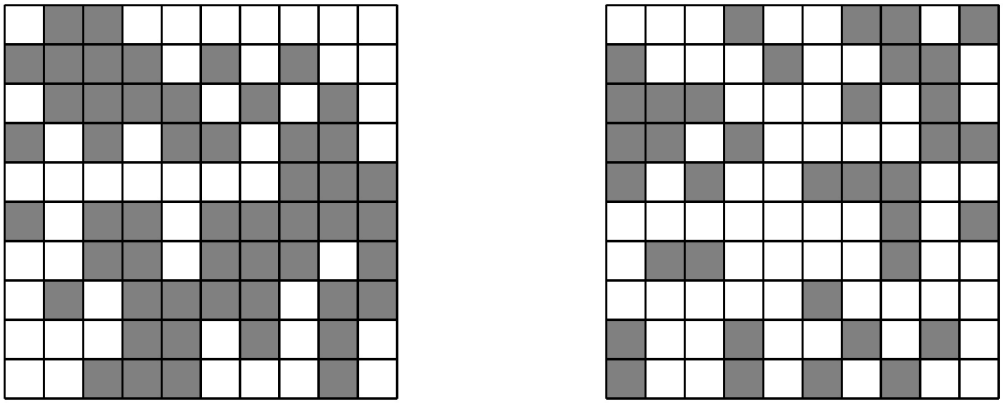

FIGURE 1 – deux exemples de grilles 10×10 ; la percolation n'est possible que dans le second cas (les cases ouvertes sont les cases blanches)

On conçoit aisément que la réussite ou non de la percolation dépend beaucoup de p : plus celle-ci est grande, plus les chances de réussite sont importantes. Nous auront l'occasion d'observer l'existence pour de grandes valeurs de n d'un seuil critique p_0 au delà duquel la percolation a toutes les chances de réussir et en dessous duquel la percolation échoue presque à chaque fois.

1. du latin percolare : couler à travers.

1.2 Création et visualisation de la grille

1.2.1 Préparation de la grille

Les deux modules essentiels dont nous aurons besoin sont les modules `numpy` (manipulation de tableaux bi-dimensionnels) et `mathplotlib.pyplot` (graphisme), qu'il convient d'importer :

```
import numpy as np
import matplotlib.pyplot as plt
```

Nous aurons aussi besoin de la fonction `rand` du module `numpy.random` (fonction qui retourne un nombre pseudo- aléatoire de l'intervalle $[0,1[$ et accessoirement de la fonction `ListedColormap` du module `matplotlib.colors` (pour choisir l'échelle chromatique à utiliser pour la représentation graphique). Ces deux fonctions seront importées directement, puisque nous n'aurons pas besoin des modules entiers :

```
from numpy.random import rand
from matplotlib.colors import ListedColormap
```

La grille de percolation sera représentée par le type `np.array`. La fonction `np.zeros((n, p))` renvoie un tableau de n lignes et p colonnes contenant dans chacune de ses cases le nombre flottant 0.0. Une fois un tableau `tab` créé, la case d'indice (i, j) est référencée indifféremment par `tab[i][j]` ou par `tab[i,j]` et peut être lue et modifiée (comme d'habitude, les indices débutent à 0). Enfin, on notera que si tab est un tableau, l'attribut `tab.shape` retourne le couple (n, p) de ses dimensions verticale et horizontale (le nombre de lignes et de colonnes, `tab` étant vu comme une matrice).

Dans la suite de ce document, on représentera une grille de percolation par un tableau $n \times n$, les cases fermées contenant le nombre flottant 0.0 et les cases ouvertes le nombre flottant 1.0.

Q 1 : Définir une fonction `Python`, `creationgrille(p, n)` à deux paramètres : un nombre réel p (qu'on supposera dans l'intervalle $[0,1[$ et un entier naturel n, qui renvoie un tableau (n, n) dans lequel chaque case sera ouverte avec la probabilité p et fermée sinon.

1.2.2 Visualisation de la grille

Pour visualiser simplement la grille, nous allons utiliser la fonction `plt.matshow` : appliquée à un tableau, celle-ci présente ce dernier sous forme de cases colorées en fonction de leur valeur.

Les couleurs sont choisies en fonction d'une échelle chromatique que vous pouvez visualiser à l'aide de la fonction plt.colorbar(). Celle utilisée par défaut va du bleu au rouge; puisque nos grilles ne contiennent pour l'instant que les valeurs 0 ou 1, les cases fermées apparaîtrons en bleu, et les cases ouvertes, en rouge.

1.2.3 Changer l'échelle chromatique

L'argument par défaut `cmap` de la fonction `plt.matshow` permet de modifier l'échelle chromatique utilisée. La fonction `ListedColormap` va nous permettre de créer l'échelle de notre choix. Puisque nous n'aurons que trois états possibles (une case pleine représentée par la valeur 0.0), une case vide représentée par la valeur 1.0 et plus tard une case remplie d'eau représentée par la valeur 0.5) une échelle à trois couleurs suffit. Vous pouvez utiliser celle-ci :

```
echelle = ListedColormap(['black', 'aqua', 'white'])
```

Q 2 : Écrire une fonction `afficher_grille(grille,nom_de_fichier)` qui prend en argument une variable grille qui correspond à une grille de percolation générée précédemment et ne renvoyant rien mais enregistrant dans `nom_de_fichier` le graphe obtenu. On pourra exporter une grille de 10×10 cases avec l'échelle suggérée précédemment, l'enregistrer sous le nom "tp09_q02_vos_noms.png" et l'envoyer à votre professeur.

1.3 Percolation

Une fois la grille crée, les cases ouvertes de la première ligne sont remplies par un fluide, ce qui sera représenté par la valeur 0.5 dans les cases correspondantes. Le fluide pourra ensuite être diffusé à chacune des cases ouvertes voisines d'une case contenant déjà le fluide jusqu'à remplir toutes les cases ouvertes possibles.

Q 3 : Écrire une fonction `percolation(grille)` qui prend en argument une grille et qui remplit de fluide celle-ci, en appliquant l'algorithme exposé ci-dessous :

1. **Créer une liste contenant initialement les coordonnées des cases ouvertes de la première ligne de la grille et remplir ces cases de liquide.**
2. **Puis, tant que cette liste n'est pas vide, effectuer les opérations suivantes :**
   (a) **extraire de cette liste les coordonnées d'une case quelconque;**
   (b) **ajouter à la liste les coordonnées des cases voisines qui sont encore vides, et les remplir de liquide.**

L'algorithme se termine quand la liste est vide.

Q 4 : Rédiger un script vous permettant de visualiser une grille avant et après remplissage, et faire l'expérience avec quelques valeurs de p pour une grille de taille raisonnable (commencer avec $n = 10$ pour vérifier visuellement que votre algorithme est correct, puis augmenter la taille de la grille, par exemple avec $n = 64$). On pourra exporter et l'enregistrer sous le nom "tp09_q04_vos_noms.png" et l'envoyer à votre professeur.

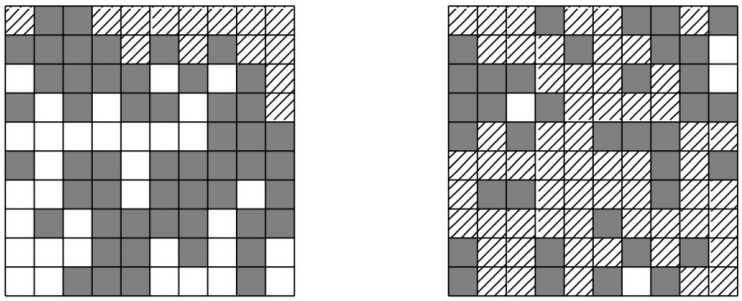

FIGURE 2 – les deux grilles de la figure 1, une fois le processus de percolation terminé (le fluide est représenté par des hachures) .

On dit que la percolation est réussie lorsqu'à la fin du processus au moins une des cases de la dernière ligne est remplie du fluide.

Q 5 : Écrire une fonction `teste_percolation(p,n)` qui prend en argument un réel $p \in [0,1[$ et un entier $n \in \mathbb{N}^*$, crée une grille, effectue la percolation et retourne :

- `True` **lorsque la percolation est réussie, c'est-à-dire lorsque le bas de la grille est atteint par le fluide ;**
- `False` **dans le cas contraire.**

1.4 Seuil critique

Nous allons désormais travailler avec des grilles de taille 128×128 [2]

Faire quelques essais de percolation avec différentes valeurs de p. Vous observerez assez vite qu'il semble exister un seuil p_0 en deçà duquel la percolation échoue presque à chaque fois, et au delà duquel celle-ci réussit presque à chaque fois. Plus précisément, il est possible de montrer que pour une grille de taille infinie, il existe un seuil critique p_0 en deçà duquel la percolation échoue toujours, et au delà duquel la percolation réussit toujours. Bien évidemment, plus la grille est grande, plus le comportement de la percolation tend à se rapprocher du cas de la grille théorique infinie.

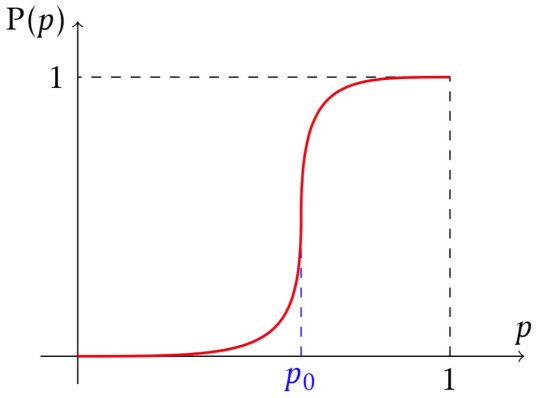

FIGURE 3 – L'allure théorique du graphe de la fonction P(p)
.

Notons $P(p)$ la probabilité pour le fluide de traverser la grille. Pour déterminer une valeur approchée de la probabilité de traverser la grille, on se contente d'effectuer k essais pour une valeur de p puis de renvoyer le nombre moyen de fois où le test de percolation est vérifié.

Q 6 : Rédiger la fonction `proba(p,k,n)` qui prend en argument le nombre d'essai k, la variable p ainsi que le nombre de cases n sur la largeur de la grille et qui renvoie $P(p)$.

Q 7 : Ecrire une fonction `tracer_proba(n,nom_de_fichier)` qui prendre en argument une taille n ne renvoyant rien mais enregistrant dans `nom_de_fichier` le graphe obtenu. On pourra traiter le cas d'une grille de 128×128 cases et enregistrer la figure obtenue sous le nom "tp09_q07_vos_noms.png" et l'envoyer à votre professeur.

2. Baisser cette valeur si le temps de calcul sur votre ordinateur est trop long.