Travaux pratiques

informatique commune

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 XX^{e} siècle et touche aujourd'hui de nombreuses disciplines, allant des mathématiques à l'économie en passant par la physique et la géologie.

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 ?

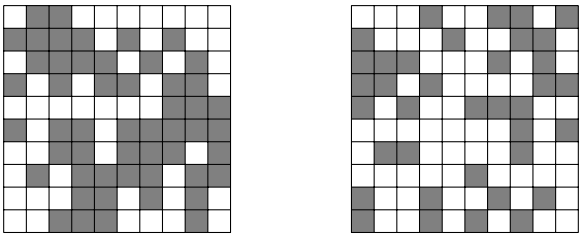

FIGURE 1 – *deux exemples de grilles* 10×10 *; la percolation n'est possible que dans le second cas (les cases ouvertes sont les cases blanches).*

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 deçà duquel la percolation échoue presque à chaque fois.

Création et visualisation 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).

1. du latin *percolare* : couler à travers.

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`.

Question 1. Définir une fonction `creation_grille(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 $n \times n$ dans lequel chaque case sera ouverte avec la probabilité p et fermée sinon.

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 [2]. 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.

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'])
```

ou celle de votre choix, dans la limite du bon goût.
(Vous trouverez la liste des couleurs prédéfinies à l'adresse `http://www.python-simple.com/img/img36.png`)

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.

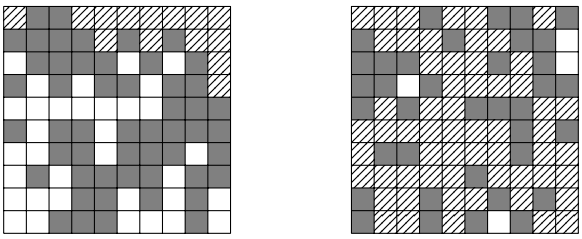

FIGURE 2 – *les deux grilles de la figure 1, une fois le processus de percolation terminé (le fluide est représenté par des hachures).*

Question 2. Écrire une fonction `percolation` qui prend en argument une grille et qui remplit de fluide celle-ci, en appliquant l'algorithme exposé ci-dessous.

(i) Créer une liste contenant initialement les coordonnées des cases ouvertes de la première ligne de la grille et remplir ces cases de liquide.

(ii) Puis, tant que cette liste n'est pas vide, effectuer les opérations suivantes :
- extraire de cette liste les coordonnées d'une case quelconque ;
- 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.

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 $n = 64$ pour vérifier visuellement que votre algorithme est correct, puis augmenter la taille de la grille, par exemple avec $n = 512$).

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.

2. Ne pas oublier l'instruction `plt.show()` pour afficher la grille si le mode interactif est desactivé.

Question 3. Écrire une fonction `teste_percolation` 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.

Seuil critique

Nous allons désormais travailler avec des grilles de taille 128×128 [3]. 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.

Question 4. Notons $\mathrm{P}(p)$ la probabilité pour le fluide de traverser la grille.

a) Proposer une démarche expérimentale simple pour estimer cette quantité en fonction de p, et rédiger la fonction `proba` correspondante.

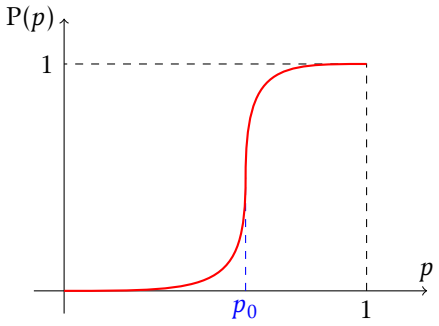

Figure 3 – *L'allure théorique du graphe de la fonction* P.

b) En utilisant une recherche dichotomique, chercher à estimer le plus précisément possible la valeur numérique du seuil p_0.

Propriétés macroscopiques

A l'instar de la thermodynamique, on peut décrire le comportement d'un système lors d'une transition de phase en introduisant des propriétés macroscopiques. Dans le cas de la percolation, on peut par exemple définir la densité moyenne $d(p)$ de cases ouvertes atteintes par le fluide.

Question 5.

a) Définir une fonction `densite` qui prend en argument une grille dans laquelle la percolation a eu lieu et qui retourne la valeur de sa densité.

b) À l'aide d'un nombre raisonnable d'échantillons, définir alors la fonction `d` qui à une probabilité $p \in [0,1[$ associe la densité moyenne de la percolation dans une grille 128×128.

c) Tracer le graphe de $d(p)$ pour $p \in [0,1[$.

Et pour les plus rapides

Recommencez toute cette étude, mais cette fois-ci avec une grille hexagonale. Il est possible de prouver que pour une grille hexagonale carrée le seuil critique p_0 est exactement égal à $p_0 = 1/2$; le vérifier expérimentalement et chercher à le démontrer (en utilisant un argument de symétrie).

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