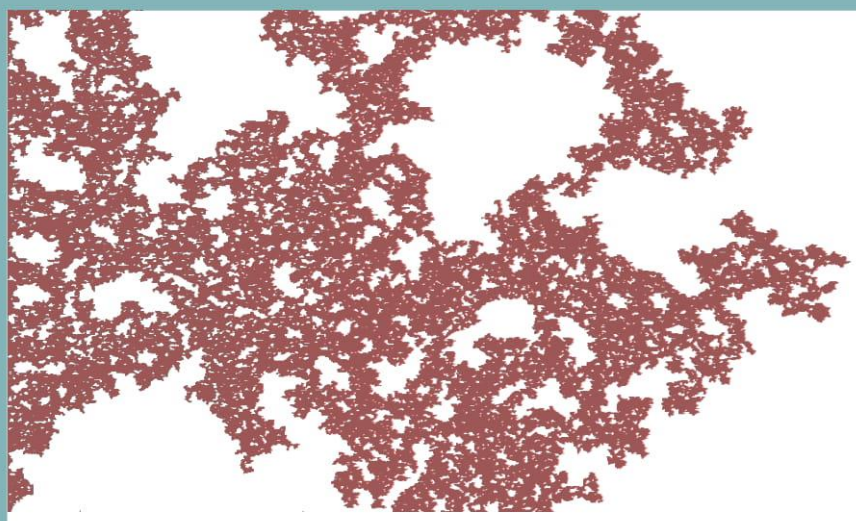


VISUALISATION DU SYSTEME DYNAMIQUE :

PERCOLATION ;
PROPAGATION DE LIQUIDE ET POROSITE DE MATIERE

Rapport ARE Dynamic

2017-2018

ENCADREURS :

GUILLON ARTHUR

LEFEBVRE BAPTISTE

ETUDIANTS :

MENZLI HELA

JRIBY JAWHAR

SOMMAIRE

Remerciements:.... 3
Introduction:.... 4
Définition de la percolation :.... 4
Usage et applications réelles:.... 4
Problématique :.... 6
Système :.... 7
Paramètre :Porosité.... 7
Modélisation de système:.... 8
Concept :.... 8
Code :.... 9
Bibliotheques:.... 9
Initialisation de la matrice:.... 9
Principe de propagation:.... 10
Simulation:.... 11
Visualisation:.... 12
Phénomène :.... 12
Conclusion: entre Résultat et attente:.... 15
Discussion:.... 16
Références:.... 18

REMERCIEMENTS :

On tient à remercier nos professeurs, ***M. Arthur Guillon et M. Baptiste Lefebvre*** qui nous ont beaucoup aidés dans l'avancement de notre recherche. On les remercie pour leur accueil, le temps passé ensemble et le partage de leur expertise au quotidien. Leur écoute et leurs conseils nous ont permis également d'atteindre finalement la bonne réponse à notre problématique.

En tant que Hela Menzli, je tiens à remercier vivement, ***M. Maximilien Danisch***, mon prof de programmation en premier semestre, qui fut d'une aide précieuse dans les moments les plus délicats et grâce à sa confiance j'ai pu me réorienter en math/informatique (en portail MIPI) et pu m'accomplir totalement dans ses disciplines (l'ARE est en une).

INTRODUCTION:

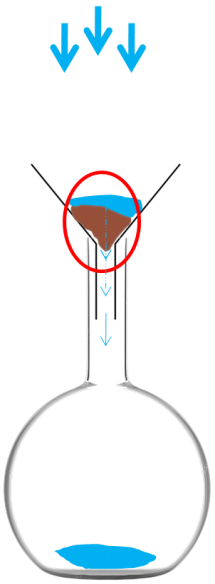
Définition de la percolation:

➢ **Littéralement:** (du latin percolare, « filtrer », « passer au travers »)

La percolation désigne communément le passage d'un fluide à travers un milieu plus ou moins perméable.[1]

➢ **Phénomène:**

La percolation a un sens plus précis en physique et en mathématiques : c'est un processus physique critique qui décrit, pour un système, une transition d'un état vers un autre. C'est un phénomène de seuil associé à la transmission d'une « information » par le biais d'un réseau de sites et de liens qui peuvent, selon leur état, relayer ou non l'information aux sites voisins.[2]



Usage et Applications Réelles :

- **Céramiques**:

Certaines céramiques ou les caoutchoucs chargés sont formés d'éléments conducteurs et isolants disposés aléatoirement. Ils sont isolants tant que la proportion de conducteurs est inférieure au seuil, où ils deviennent

brusquement conducteurs. Leur conductivité σ varie ensuite comme (p − pC)t, t étant un exposant critique de transport (à d= 2, t = 1,3). L'étude de la réponse en fréquence de ces milieux est fondée sur le caractère fractal de l'amas infini conducteur. Le comportement des céramiques parafoudres relève de la percolation dirigée.

- **Gélification:**

Dans la transition de gélification (la prise d'une gélatine par exemple), des éléments polymériques s'associent pour former un réseau tridimensionnel. Au seuil de gélification (correspondant au seuil de percolation) apparaît un amas infini (le gel). On associe à la masse du gel la notion de paramètre d'ordre (exposant β), l'élasticité du gel pouvant être reliée à l'exposant t de la conductivité.

- **L'exemple le plus fameux/simple à expliquer: Percolation de café:**

Lorsqu'on verse de l'eau chaude sur du café moulu L'eau doit se frayer un chemin parmi les mini-canaux créés par les grains de café, après avoir « imbibé » chaque grain.

Selon les conditions expérimentales : taille des grains, compactage plus ou moins fort du café, pression de l'eau, forme du filtre,... l'eau arrivera à traverser le café ou pas : si elle y arrive, le café « passe » et l'on dit que l'on a atteint le seuil de percolation.

PROBLÉMATIQUE:

D’après l’étude de phénomène on a pu sélectionner la problématique suivante qui nous permettra d’étudier le dynamisme de notre système:

COMMENT ÉVOLUE LA PROPAGATION DU FLUIDE PAR RAPPORT À LA POROSITÉ DE LA MATIÈRE?

SYSTÈME:

Avant la modélisation de tout système on a besoin de préciser ses paramètres qui agissent sur son dynamisme.

Paramètre :Porosité

La théorie de la percolation est une branche de la physique statistique et mathématique qui s'intéresse aux caractéristiques des milieux essentiellement ***poreux***.

- **Porosité Définition[3] :**

  La porosité est l'ensemble des vides (pores) d'un matériau solide, ces vides sont remplis par des fluides (liquide ou gaz). C'est une grandeur physique entre 0 et 1 ou en pourcentage entre 0 et 100%, qui conditionne les capacités d'écoulement et de rétention d'un substrat (voir aussi Loi de Darcy).

  La porosité est aussi une valeur numérique définie comme le rapport entre le volume des vides et le volume total d'un milieu poreux :

$$\phi = \frac{V_{pores}}{V_{total}}$$

Ainsi on a choisit comme paramètre pour notre système la porosité de la matière et on l'a noté p :

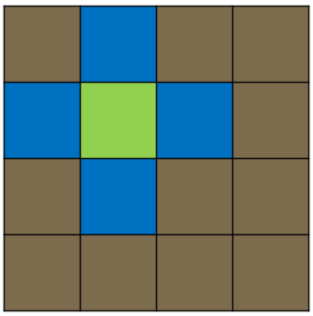
```
In [41]: p= 0.76 ## Cas du sable N5
         N=100
```

Modélisation de système:

Concept:

La situation peut être modélisée par un réseau carré bidimensionnel dans lequel des sites individuels peuvent être occupés par le milieu ou ils peuvent être vides. La porosité p est la fraction des sites vides.

Le liquide peut traverser le milieu à travers les sites vides adjacents. Par adjacente nous entendons les 4 prochains voisins (haut-gauche-bas-droite) d'un site dans un réseau carré.



Demander "Est-ce que le liquide fait tout le chemin vers le bas?" est alors équivalent à demander "Y a-t-il un chemin de sites vides adjacents qui relie le haut et le bas?". Il se trouve que c'est une question facile qui, cependant, est très difficile à répondre. Le système particulier ci-dessus (réseau carré 2-d avec 4 voisins) a une porosité critique pc=0.592746 Cette porosité correspond aux lois suivantes:

- ***Si P < Pc:*** le liquide n'atteint jamais le fond.
- ***Si P > Pc:*** il l'atteint, d'où la notation Théorie de percolation.

Code:

- ***Bibliothèques:***

On a commencé notre code, après avoir déclaré les paramètres, par l'appel des bibliothèques python qu'on va utiliser lors de nos simulations :

```
In [42]: # %matplotlib inline
         import numpy as np
         from numpy import random
```

Numpy : avec la quelle on a exploité les matrices et pu générer des valeurs aléatoires pour son remplissage.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
%matplotlib notebook
```

On a utilisé aussi la bibliothèque graphique ***matplotlib*** pour les visualisations.

- ***Initialisation :***

On modélise le système par une matrice de taille N*N qu'on l'initialise d'abord par des 1 qui représentent les obstacles puis on lui affecte aléatoirement p*(N*N) 0 qui représentent les pores (avec p notre paramètre de porosité).

On a écrit ainsi la fonction ***init_mat()*** qui traduit ce qu'on a dit ci-dessus:

```python
def init_mat():
    m=np.ones((N, N))
    i=0
    j=0
    nb= int(p*N*N)
    positions = [(i,j) for i in range(N) for j in range(N)] # Produit cartésien [0,100[*[0,100[
    positions = np.array(positions)
    indices = np.random.choice(np.arange(len(positions)), nb, replace=False)
    for k in range(nb):
        i=positions[indices][k][0]
        j=positions[indices][k][1]
        m[i,j]=0
    return m
```

- ***Principe de propagation :***

Pour décrire la propagation de liquide on a commencé, au niveau code, par faire tourner la fonction ***propage_liquide*** qui prend en paramètres une matrice (initialisée par la fonction init_mat()) et teste ses cases, case par case, si elles sont des obstacles, déjà remplies par le liquide ou encore vides:

```python
def propage_liquide(matrice):
    (a,b) = matrice.shape
    new_mat = np.zeros((a,b))
    for i in range(a):
        for j in range(b):
            if matrice[i,j] == 1:
                new_mat[i,j] = 1
            elif matrice[i,j] == 2:
                new_mat[i,j] = 2
            else:
                if len(voisins_pleins(matrice, i, j)) > 0:
                    new_mat[i,j] = 2

    return new_mat
```

Ainsi si une case présente un obstacle ou déjà remplie on la garde dans la nouvelle copie « vide » de la matrice si non on fait appel à la fonction ***voisins_pleins*** qui prend en paramètres l'ancienne matrice et les indices de cette case :

```python
def voisins_pleins(m, i, j):
    a,b = m.shape
    cases = []
    if i < a - 1 and m[i+1, j] == 2:
        # remplir case au dessous
        cases.append((i+1, j))
    if i > 0 and m[i-1, j] == 2 :
        # remplir case au dessus
        cases.append((i-1, j))
    if j < b-1 and m[i, j+1] == 2:
        # remplir à droite
        cases.append((i, j+1))
    if j > 0 and m[i, j-1] == 2:
        # remplir à gauche
        cases.append((i, j-1))
    return cases
```

Cette fonction teste si la case a des voisins déjà pleins par le liquide et les affecte à une liste s'ils existent (cas pour le quel on teste sur la longueur de la liste dans propage_liquide). La taille maximale qu'une liste pourra avoir est ainsi 4, remplie par [Voisin rempli au dessous,Vaudessus,Vàdroite, Vàgauche]et le minimum est 0 (cas d'aucun voisin rempli).

- ***Simulation:***

Pour tester les précédentes fonctions on a écrit notre dernière fonction ***simulation()***:

```python
def simulation():
    new_mat = init_mat()
    (a,b) = new_mat.shape
    for j in range(b):
        new_mat[0, j] = 2
    results = [new_mat.copy()]
    matrice = np.zeros_like(new_mat)
    while (new_mat != matrice).any():
        matrice = new_mat
        #start_time = time.time()  # s
        new_mat = propage_liquide(matrice)
        #end_time = time.time()  # s
        # print("{} s".format(end_time - start_time))
        results.append(new_mat.copy())
    return results
```

On a commencé par remplir la première ligne de la matrice par des 2 (en tant que liquide versé sur la surface de la matière).

Apres on s'est servit, pour étudier la progression de la propagation du liquide, par une comparaison de deux matrices (initialisées chacune comme décrit ci-dessus : avec décalage d'étape), assurée par une boucle "while" (inspirée par le principe de la tri à bulle).

- ***Visualisation:***

Finalement, pour pouvoir visualiser les résultats de nos simulations on a introduit ce code:

```
results = simulation()
```

```
# from pylab import *
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
%matplotlib notebook

# size = np.array(results[0].shape)
# dpi = 300.0
# figsize= size[1]/float(dpi),size[0]/float(dpi)
# fig = plt.figure(figsize = figsize, dpi = dpi, facecolor = "white")
fig = plt.figure()
im = plt.imshow(results[0], interpolation = 'nearest', cmap = plt.cm.gray_r, animated=True)
plt.xticks([]), plt.yticks([])

def update(i):
    im.set_array(results[i])
    return im,


ani = animation.FuncAnimation(fig, update, frames=range(0, len(results)), interval=200, repeat=True)

plt.show()
ani.save('testa76%.gif', fps=30, extra_args=['-vcodec', 'libx264'])
```

On obtiendra ainsi des visualisations différentes qu'on va les étudier dans la partie suivante.

- ***Phénomène :***

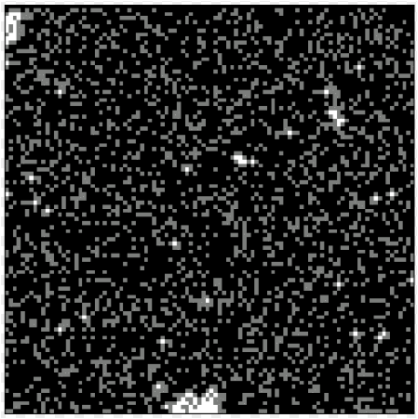
D'après l'étude des paramètres/selon la porosité de la matière, on va visualiser la percolation des différents cas possibles:

➢ Système dont la porosité est supérieure à la porosité critique p_c

- Système dont la porosité est inférieure à la porosité critique p_c
- Système dont la porosité est égale à la porosité critique p_c

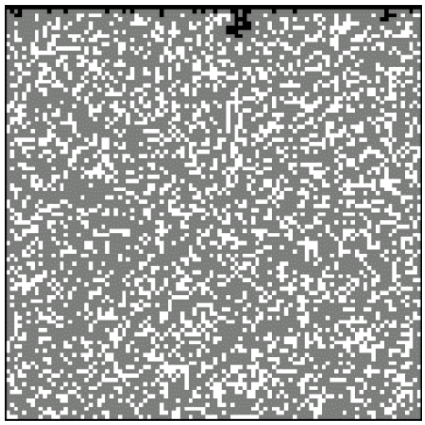
Ps: dans les figures qui suivent le noire correspond au liquide, le blancs au pores (vides) et le gris aux cases obstacles.

1ER CAS : P > PC:



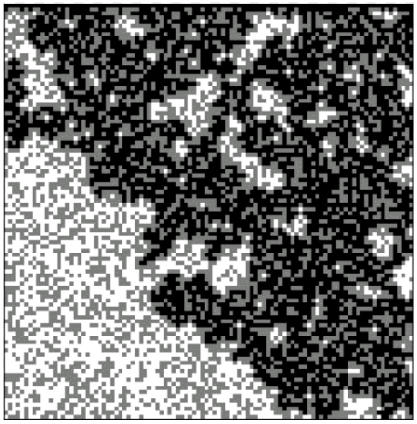
Propagation de liquide importante: Liquide atteint le fond.

2EME CAS : P < PC:



Propagation de assez faible (presque nulle/ variation négligeable).

3EME CAS : P=PC :



Pour une même valeur de p et pour deux simulations successives on eu ces deux différents cas.

=> D’après notre système : pour une porosité égale à la porosité critique on ne sait jamais si le liquide va atteindre le fond ou pas.

CONCLUSION ENTRE RESULTAT ET ATTENTE:

Après ce qu'on a eu en visualisation on constate que:

- **Variable de système :**

  Nos résultats sont confondus avec la réalité au niveau de l'influence de la porosité de la matière sur la variation de la propagation de liquide

- **Lois physiques agissantes sur le système :**

  - Visualisation qui respecte la continuité de liquide.
  - Visualisation qui, dans certains cas, ne respecte pas la loi de gravité qui détermine un sens de propagation précis (du haut vers le bas)

DISCUSSION :

Après l'étude des autres principes de propagation qui respectent le sens de propagation comme celui-ci-dessous

```
def remp_par_liquide(m):
    a, b = m.shape
    results = []
    i = 0
    for j in range(b):
        m[i, j] = 2
    results.append(m.copy())
    for i in range(0, a):
        for j in range(b):
            # remplir les cases voisines
            if m[i, j] == 2:
                m = voisin(m, i, j)
            if m[i, (a - 1) - j] == 2:
                m = voisin(m, i, (a - 1) - j)
            results.append(m.copy())
    return results
```

On a finit par déterminer leurs limites (par en trouver des contres exemples qui montrent qu'ils s'opposent à la loi de continuité de liquide)

```
def init_mat_bis(nb_rows=10, nb_cols=10):
    shape = (nb_rows, nb_cols)
    m = np.zeros(shape)
    for j in range(0, nb_cols):
        m[1, j] = 1
    m[1, 4] = 0
    return m
```

Ainsi, on a fini par écrire le code de dernier principe (mentionné dans la partie système) qui arrive, d'abord, à visualiser le dynamisme du système et respecte ensuite la continuité du liquide, mais qui, pour des cas trop particuliers, ne respecte pas la loi de gravité qui détermine un sens précis de propagation (de haut vers le bas).

Ainsi, malgré qu'on a pu répondre à notre problématique avec le code qu'on a eu, on a proposé une amélioration qui consiste à fusionner les deux codes pour obtenir à la fin un code qui sera confondu avec la réalité en tout point.

REFERENCES :

Définitions :

Percolation [1] : D'après Wikipédia

Percolation [2] : D'après Wikipédia

Porosité [3]: D'après Wikipédia

Sources schémas :

Notre propre création avec les outils d'insertion de Microsoft office.

Sources de recherche :

- Chaine youtube de *KClassScienceChannel* : https://www.youtube.com/watch?v=N_pARn50GHo
- Encyclopédie Wikipédia
- Encyclopédie *Universalis* : https://www.universalis.fr/encyclopedie/percolation/4-quelques-applications/
- Notebooks Outil et bibliothèques de langage python proposé dans le Github par les encadreurs d'ARE : https://github.com/Helaa/are-dynamic-2017
- Site *Complexity Explorables* proposé par le moodle : http://rocs.hu-berlin.de/explorables/explorables/0.592746/