Rapport de projet de simulation numérique
Malin comme un...blob

Le 27 Mai 2020

Groupe 4

ALQUIER Rodolphe
BALANNEC Ancelin
CAVALIER Pierre
CONTENTIN Théo
HARDY Paul

Université Paris-Saclay
L2 Double Licence Maths-Physique

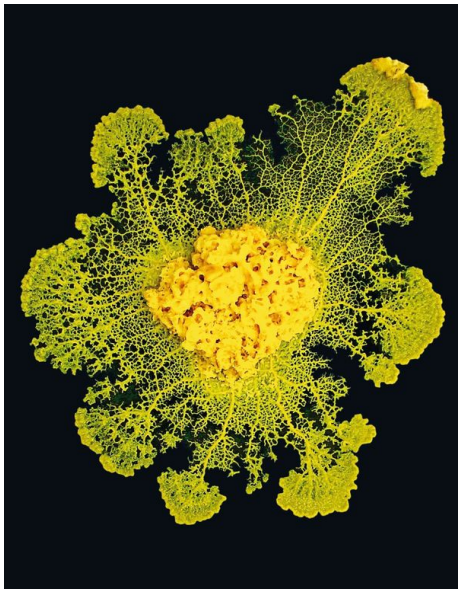
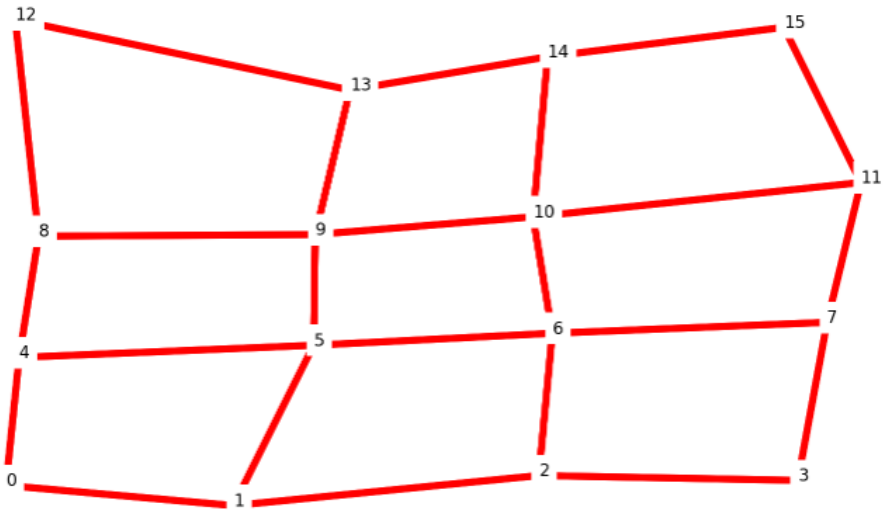


Table des matières

1 Introduction

Durant ce projet, nous avons travaillé sur "le blob" ; cet organisme unicellulaire qui peut croître de plusieurs centimètres par heure et devient donc une cellule macroscopique. Le *Physarum polycephalum*, de son vrai nom, peut créer des structures internes, des ramifications permettant d'acheminer la nourriture des sources aux puits. Dans ce rapport, nous avons cherché à modéliser un blob de la manière la plus simple possible : des points (ou nœuds) et des branches (ou ramifications). Nous avons fait varier plusieurs paramètres afin de balayer le plus de dispositions possibles. Pour identifier les nœuds, nous les avons numérotés de 0 à $N-1$ avec N le nombre total de points (N est un carré d'entier pour le maillage rectangulaire) de cette manière :



2 Présentation du modèle

On suppose qu'à l'état initial le blob est un réseau de N nœuds tous équivalents numérotés de 0 à $N-1$. Ces nœuds sont reliés entre eux par des branches dans lesquelles circuleront les nutriments. Les branches ont des longueurs et diamètres (ou épaisseurs) variables . Le débit Q_{ij} dans la branche reliant le nœud i au nœud j est donné par la loi de Poiseuille :

$$Q_{ij} = \frac{\pi r^4}{8\eta L_{ij}}(p_j - p_i) = \frac{D_{ij}}{L_{ij}}(p_j - p_i) \tag{1}$$

où, L_{ij} est la longueur de la branche, η est la viscosité du liquide, p_i et p_j les pressions au niveau des nœuds i et j. Au niveau du nœud i, le débit est positif lorsque le fluide va vers celui-ci. Comme la viscosité est la même partout, on introduit la conductivité D_{ij}.

Si le nœud k est une source, $\sum_j Q_{kj} = I_k < 0$. Ici I_k est le débit total sortant du nœud. De même, si le nœud l est un puits, $\sum_j Q_{lj} = I_l > 0$. Pour les autres nœuds i, la conservation du débit s'ecrit : $\sum_j Q_{ij} = I_i = 0$.

Enfin, tout ce qui rentre dans le réseau doit en ressortir. Ceci conduit à la relation : $\sum_k I_k + \sum_l I_l = 0$.

On remarque qu'il est possible de déterminer les pressions p_i de tous les nœuds en résolvant un système de N équations à N inconnues, ce qui donne $AP = I$ avec A une matrice $(N \times N)$, P un vecteur à N dimensions où la i-ème ligne correspond à la pression p_i , et I est un vecteur à N dimensions dont la i-ème ligne correspond au débit total I_i au niveau du nœud i.
En résolvant ce système on obtient toutes les pressions, et ainsi on peut calculer les débits Q_{ij} dans chaque branche reliant le nœud i et j.

Le rayon (ou épaisseur) r de chaque branche évolue au cours du temps. Il tend à diminuer avec le temps si le débit de nutriments le traversant est faible. En revanche si un débit Q_{ij} non nul s'écoule dans la branche, son rayon est non seulement stabilisé, mais il sera d'autant plus grand que le débit sera élevé (jusqu'à un certain point).

La conductivité D_{ij} évolue avec le temps t de la manière suivante :

$$\frac{dD_{ij}}{dt} = f(Q_{ij}) - \frac{D_{ij}}{\tau} \tag{2}$$

Avec τ le temps caractéristique de disparition d'une branche et f une fonction modélisant la réponse de la branche au débit la traversant :

$$f(Q) = \frac{|Q|^\gamma}{1 + |Q|^\gamma} \tag{3}$$

Et pendant un temps $dt << \tau$ nous obtenons :

$$D_{ij}(t + dt) \approx D_{ij}(t) + (f(Q_{ij}) - \frac{D_{ij}}{\tau})dt \tag{4}$$

3 Résolution mathématique AP = I

Comme expliqué ci-dessus on cherche la matrice A qui nous permettra d'obtenir P la matrice correspondant aux pressions de chaque nœud.

$$\begin{pmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{pmatrix} \times \begin{pmatrix} p_1 \\ \vdots \\ p_N \end{pmatrix} = \begin{pmatrix} I_1 \\ \vdots \\ I_N \end{pmatrix} \tag{5}$$

Où

$$I_i = \sum_{\text{j}} Q_{ij} \tag{6}$$

Or Q_{ij} est nul si i et j ne sont pas voisins, on note ainsi $V[i]$ le voisinage de I et ainsi :

$$I_i = \sum_{j \in V[i]} \frac{D_{ij}}{L_{ij}} (p_j - p_i) \tag{7}$$

On obtient finalement :

$$a_{ij} = \begin{cases} -\sum\limits_{j \in V[i]} \frac{D_{ij}}{L_{ij}} & \text{si } j = i \\ \frac{D_{ij}}{L_{ij}} & \text{si } j \in V[i] \\ 0 & \text{si } j \notin V[i] \end{cases} \tag{8}$$

Dans notre cas, la matrice A est de rang $N-1$ et pour résoudre ce système, on utilise la méthode des moindres carrés avec le module **np.linalg.lstsq** de Numpy.

4 Résolution informatique

Pour répondre au problème qui nous a été posé nous avons choisi de travailler avec deux classes, une pour les nœuds du réseau et une pour le blob en lui même.

La classe des noeuds nous permet de répertorier toutes les informations sur un nœud. À un point n on associe ses coordonnés x et y comme vu précedemment et une légère fluctuation aléatoire dans un carré de côté 0.5 et de centre le point. À ce point, on associe sa liste de voisins (create_voisins) et à partir de cette dernière on obtient la distance par rapport aux voisins (get_L) et la conductivité (get_D) par rapport à ses voisins.

À partir de cette classe, nous avons créer la classe blob qui génère tout le maillage avec des options qui permettent différents type de situations ainsi qu'une répartition des puits et des sources aussi bien constante que variable. De plus cette classe initialise aussi les paramètres des nœuds ainsi que l'intensité des sources et des puits.

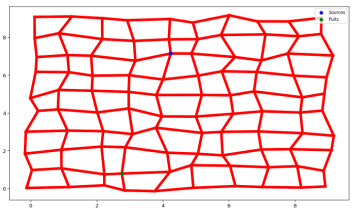
De plus, nous avons rajouté une fonction pour répresenter graphiquement chaque partie du blob (plot (plot3D dans le cas à trois dimensions)), que ce soit les nœuds (plot_noeud), les branches (plot_branch), les puits (plot_puits), les sources (plot_sources). Nous avons ajouté une fonction (get_noeud) qui renvoie l'objet nœud correspondant à l'indice n donné, ainsi que des fonctions qui calculent la matrice A comme vu précedemment dans la partie 3, qui permettent d'obtenir l'intensité (get_I) en un nœud et qui calculent la pression (calcule_p) en un nœud donné.

Ensuite nous avons la fonction (calcule_D) qui permet d'avancer à l'instant $t+dt$, à partir des données à l'instant t en calculant la conductivité à partir de l'équation (4). Pour cette fonction nous avons dû résoudre une équation différentielle(equa_diff).

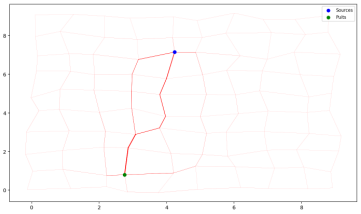
Nous avons, en outre, créé des sous-classes de la classe noeud dont le maillage est différent (hex-noeud, Delaunoeud...) pour que ces sous-classes héritent des attributs de nœud tout en ayant la possibilité de recoder la fonction create_voisins qui ne marche plus quand on sort du maillage rectangulaire.

5 Observations et analyse des résultats

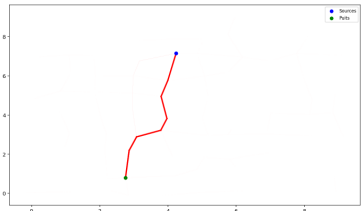
Nous avons donc testé notre blob dans la forme la plus simple, avec un puit et une source fixés. Avec $\tau = 1$ et $\gamma = 1.8$ et un pas de temps $dt = 0.01$ nous avons lancé la simulation ce qui donne l'évolution ci-dessous :



Situation initiale



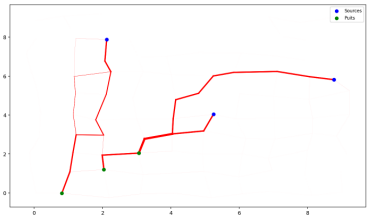
Après 500 itérations

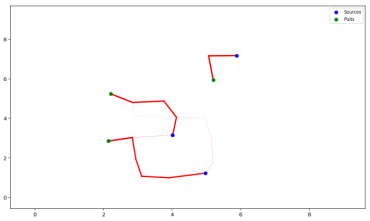


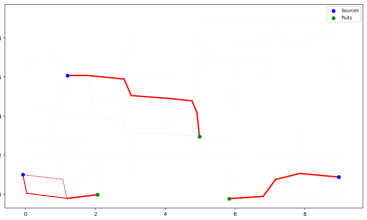
Après 1000 itérations

La simulation est donc cohérente avec la théorie, le blob crée un chemin optimisé entre le puit et la source, la dernière branche restante correspond au plus petit chemin reliant la source au puit.

Rajoutons plusieurs puits et sources à différents endroits (toujours fixes dans le temps) et analysons les résultats après 1000 itérations. Les trois images résultantes sont donc issues de situations initiales distinctes.

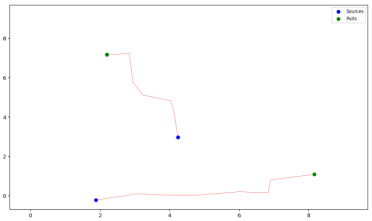




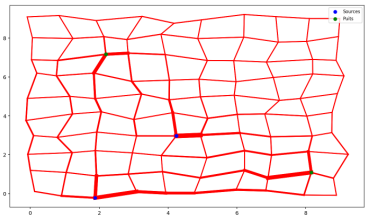


On remarque sur ces trois exemples que la connexion entre les puits et les sources est optimale, il n'y a pas de manière plus courte de relier les sources aux puits que celle effectuée par le blob.

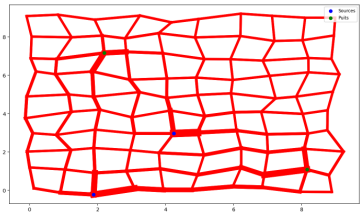
Nous allons désormais nous intéresser à l'impact de τ et de γ sur l'évolution du blob. Nous avons fait varier la valeur de τ entre 0.1 et 100. Voici ce que nous avons observé :



$\tau = 0.1$



$\tau = 10$

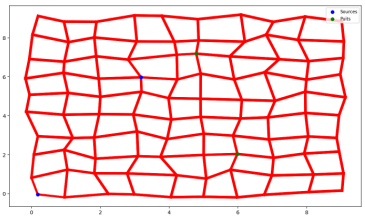


$\tau = 100$

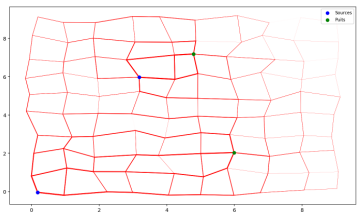
Les graphes représentent respectivement l'évolution du blob à $\tau = 0.1$, 10 et 100 après 1000 itérations. On observe que plus la valeur de τ est petite, plus l'état final (l'état dans lequel il ne reste que les chemins optimisés entre les sources et les puits) est atteint rapidement. Ce résultat était prévisible puisque la dérivée

temporelle de la conductivité dépend de $-\frac{1}{\tau}$. Nous voyons également que les branches au niveau des puits et des sources sont plus épaisses quand τ augmente.

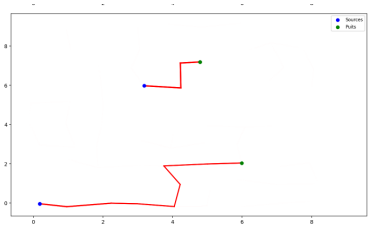
Nous avons étudié l'influence de γ sur l'évolution du blob. Voici nos résultats :



$\gamma = -1.8$



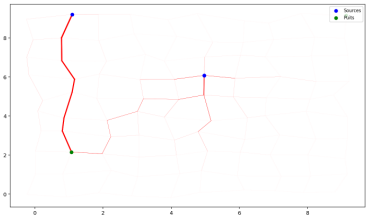
$\gamma = 0.5$



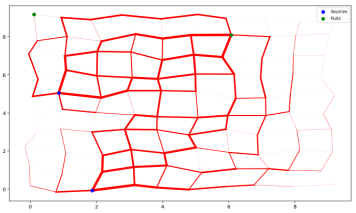
$\gamma = 5$

Comme précédemment pour τ nous avons étudié la forme finale du blob après 1000 itérations pour $\gamma = -1.8$, 0.5 et 5. On note que la vitesse de croissance augmente lorsque γ augmente ce qui est logique étant donné que plus γ est élevé plus $f(Q)$ le sera aussi et la dérivée temporelle de la conductivité dépend de $f(Q)$.

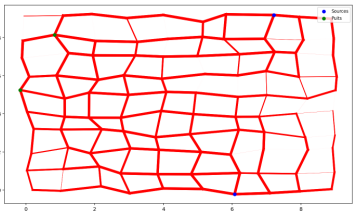
On cherche maintenant ce qu'il se passe lorsque l'on impose un débit au réseau.



$I_{source} = -1$ et $I_{puit} = 1$



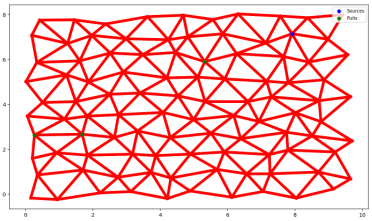
$I_{source} = -10$ et $I_{puit} = 10$



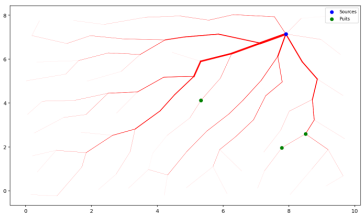
$I_{source} = -50$ et $I_{puit} = 50$

Les 3 graphes sont les résultats obtenus après 700 itérations. La première chose que l'on remarque est que plus le débit total (l'intensité) imposé est grand, plus le nombre de branche ayant un flux élevé qui les traverse est grand. Le résultat est le même quand les sources sont immobiles et les puits mobiles ce qui est logique puisque les branches seront aussi larges dans les deux sens de parcours de la nourriture. Nous avons aussi réitérer l'expérience avec des intensités variables sur chaque puits et chaque sources. Il en résulte que les branches seront d'autant plus épaisses (r sera grand) au niveau d'un point que l'intensité, en valeur absolue, sera grande en ce point.

Nous avons développé un maillage triangulaire qui fonctionne selon toutes les règles citées précédemment mais où les polygones formés sont des triangles et chaque point -hormis ceux sur les bords- a six voisins. Nous avons voulu voir ce qu'il se passerait si l'on fixait une source et qu'à chaque pas de temps, la position des puits variait. Voici le résultat pour une source et trois puits avec un maillage triangulaire :



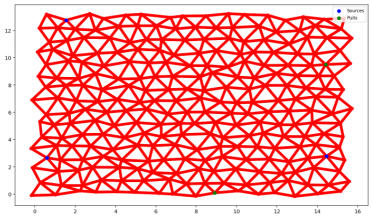
Blob initial



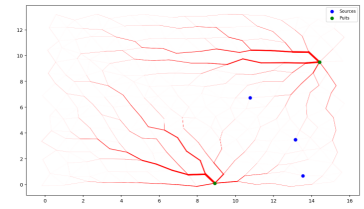
Blob après 1000 itérations

Des scientifiques japonais avaient réussi à montrer que le blob était capable de retrouver la structure du réseau du chemin de fer japonais. Nous pouvons voir que notre simulation illustre ce comportement. Imaginons que la source (point bleu) correspond à une grande ville comme Paris. Dans cette analogie, les branches représentent les routes reliant Paris aux villes voisines. Plus une branche est épaisse, plus la route est empruntée par les voyageurs. Autrement dit, plus une branche est épaisse, plus la probabilité qu'un puit s'y trouve à proximité est élevée. Ainsi nous voyons un réseau routier optimisé qui permet de relier Paris aux villes voisines.

On se retrouve avec globalement le même réseau si on fixe les puits et non plus les sources ce qui est logique étant donné que les mêmes chemins se développent, même si les acheminements se font dans le sens inverse, la conductivité ne change pas.

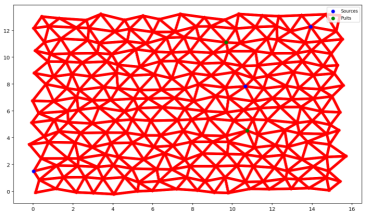


Blob initial

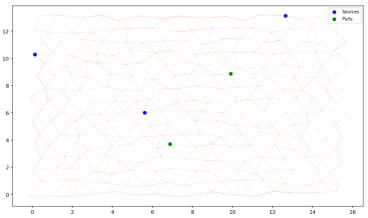


Blob après 1000 itérations

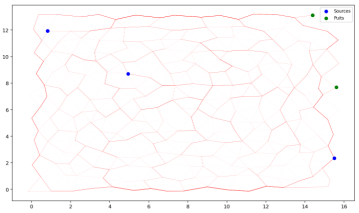
Dans la simulation suivante, nous avons fait varier les positions des puits et des sources à chaque pas de temps :



Situation initiale



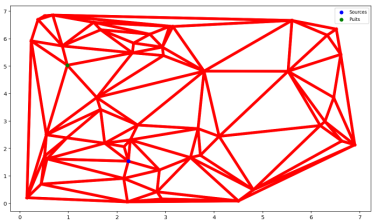
Après 500 itérations



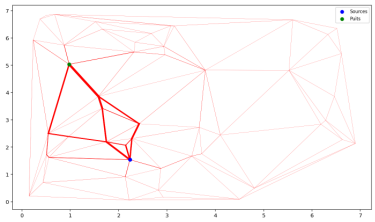
Après 1000 itérations

On observe que dans le cas où les sources et les puits ne sont pas fixes, la conductivité de tous les chemins décroît jusqu'à atteindre un certain stade à partir duquel le blob sélectionne quelles branches il délaisse et lesquelles il garde. Le blob se retrouve donc avec un maillage beaucoup plus espacé et un réseau périphérique s'installe. Étant donné que les puits et les sources sont mobiles, le blob ne garde que les chemins "utiles" permettant de relier facilement deux points alétoirement placés sur les nœuds.

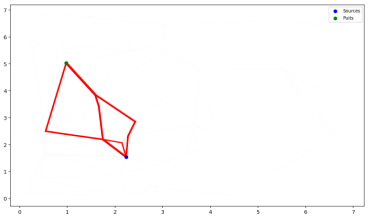
Nous avons par la suite fait une triangulation de Delaunay à partir d'un nuage de points aléatoire. On utilise la marche de Jarvis qui permet d'obtenir un polygone convexe qui englobe le nuage de point à partir de points de ce dernier. Ensuite on obtient une triangulation quelconque grâce à la méthode des oreilles à partir de cette dernière on procède à un basculement ce qui permet de transformer deux triangles avec un angle obtus qui forment un quadrilatère, avec donc deux sommets en communs, en deux triangles avec uniquement des angles aigus respectant ainsi les conditions de Delaunay.



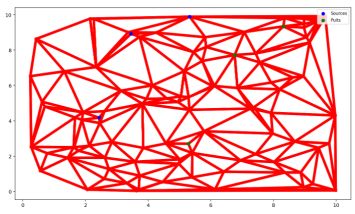
Situation initiale



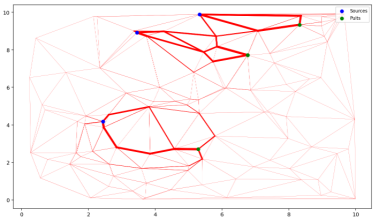
Après 500 itérations



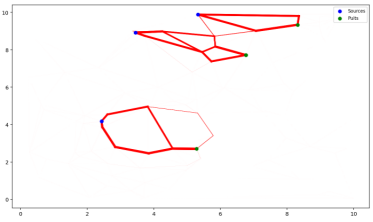
Après 1000 itérations



Situation initiale



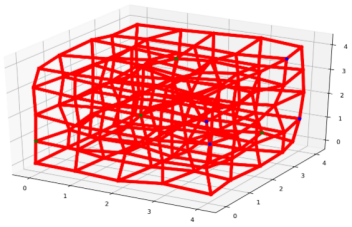
Après 500 itérations



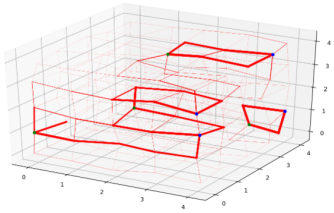
Après 1000 itérations

Sans grande surprise, le blob a le même comportement que sur un maillage rectangulaire : il cherche à relier les sources et les puits de manière optimale.

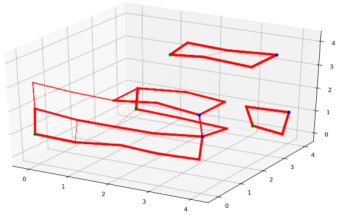
Nous avons essayé avec un autre maillage, en trois dimensions cette fois, que l'on appellera maillage cubique. Ce maillage est relativement différent des autres car on voit bien que pour optimiser ses trajets le blob reste sur la même stratégie qu'avant (à savoir un jeu de branche reliant tous les puits et sources deux à deux de manière optimale). Cependant on observe des branches annexes qui mettent beaucoup plus de temps à s'effacer.



Situation initiale



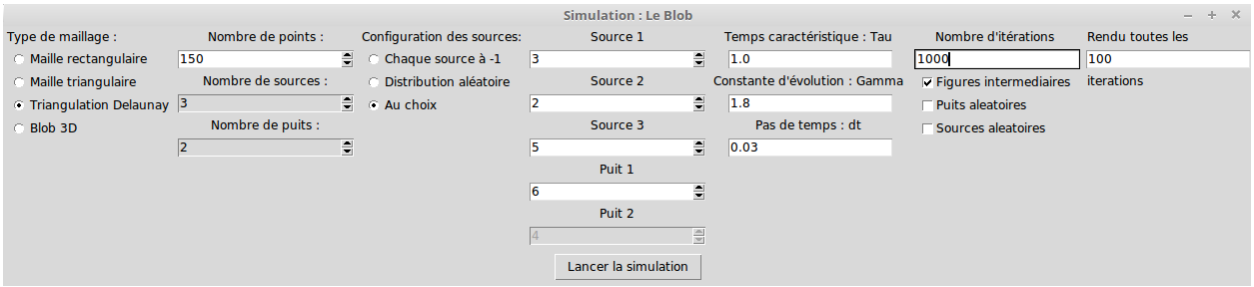
Après 500 itérations



Après 1000 itérations

6 Éxecution du code

Pour faire tourner les simulations, nous avons codé une interface graphique avec le module **tkinter** de Python. Elle consiste en une liste de boutons correspondant à chaque paramètre de la simulation. La recherche de documentation pour ce module nous ayant posé particulièrement problème.



Malheureusement cette interface ne fonctionne pas si le code est exécuté sur serveur, il faut donc se contenter d'appeler les fonctions avec leur paramètres : la documentation étant fournie dans le fichier .ipynb. Pour ouvrir l'interface il faut donc travailler avec le fichier .py en local.

7 Conclusion

À travers ce projet nous avons pu simuler informatiquement le comportement d'un blob en présence de sources et de puits de nourriture et ainsi nous rendre compte que ce dernier choisissait stratégiquement les branches du maillage à entretenir et celles à délaisser. À travers les nombreux maillages étudiés, nous avons pu constater que le blob arrivait à reproduire ce comportement quelque soit le maillage initial. Cependant lorsque les sources et/ou les puits changeaient de place à chaque pas de temps le blob développait des embranchements en ne gardant actives que les routes "utiles". Les paramètres de l'expérience, τ et de γ, n'influent que sur la vitesse de l'expérience et non pas sur le résultat final. Les résultats obtenus montrent cependant que le blob est capable, par exemple, de trouver le moyen de transport optimal entre un nombre

donné de point de départ (source) et de point d'arrivée (puit). En matière d'optimisation on pourrait aussi simuler un trafic ferroviaire en modifiant l'intensité des point d'arrivée en fonction de la population d'une ville ce qui permettrait de savoir combien de trains il faudrait envoyer par heure dans certaines gares. Évidemment le blob trouve son utilité dans les problèmes impliquant des réseaux avec des débits que ce soit en hydrostatisque ou en électrocinétique dans des circuits fermés avec des sources de courants et des "puits"/annihilateur de courant. En guise d'ouverture, il serait possible d'étendre le problème du blob à des dimensions plus élevées ou de travailler avec des sources à l'extérieur du blob et de rajouter la possibilité au blob de déplacer ses nœuds moyennant un coût en énergie.

Los Commandos