UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MODÉLISATION PAR AUTOMATES CELLULAIRES DE BRÈCHES HYDROTHERMALES

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

MARTIN LALONDE

MARS 2006

REMERCIEMENTS

Renan Furic, Adrien Pouradier et Guillaume Matton – Pour toutes les discussions, les explications et les encouragements. Je n'aurais pas pu me débrouiller sans eux dans le merveilleux monde de la géologie.

Marc-André, Sarah, Carl, François, Sylvain R., Sylvain T., Gabrielle et les autres que j'oublie – Pour les discussions et les rencontres après le travail.

Ma famille et mes proches – Pour tous leurs encouragements, leur appui moral, leur appui financier, merci d'avoir été dans ma vie et d'avoir cru en moi.

Éric Marcoux, Adrien Pouradier – Pour leur accueil lors de mon stage à Orléans, France.

Professeur Bernard Guy, Département de Géologie, École Nationale Supérieure des Mines, St-Étienne – Pour son accueil bien sûr, mais aussi pour les discussions et les réflexions qu'il a engendrées.

Professeur Michel Jébrak, Département des Sciences de la terre et de l'atmosphère, UQAM – Pour son énergie, sa simplicité, ses innombrables idées, les discussions, son appui financier, son projet de recherche incroyablement «hors des sentiers battus», et l'étincelle qu'il a fait naître en moi pour la recherche scientifique, merci de tout coeur !

Professeur Guy Tremblay, Département d'Informatique, UQAM – Pour avoir bien voulu embarquer dans un projet hors de l'ordinaire (première collaboration entre le Département d'Informatique et le Département des Sciences de la terre et de l'atmosphère), pour ses encouragements, pour son exceptionnelle rigueur, l'encadrement exemplaire qu'il m'a offert, son professionalisme et sa disponibilité, un énorme merci. Je ne saurais être plus reconnaissant.

J'ai vraiment eu beaucoup de chance d'avoir pu être encadré par deux aussi bons professeurs et directeurs de maîtrise. Merci encore mille fois ! Je suis très heureux d'avoir fait ma maîtrise à l'UQAM et j'en garderai un souvenir inoubliable.

Table des matières

Table des figures

Liste des tableaux

Résumé

Une brèche est un ensemble de blocs anguleux noyés dans un ciment de nature variable. Les brèches hydrothermales sont générées par un processus de fracturation, de dissolution des fragments, ainsi que des changements de composition causés par des eaux souterraines sous pression à haute température. La nature de la majorité des processus impliqués dans la formation des brèches hydrothermales est bien comprise d'un point de vue géochimique et plusieurs modèles basés sur cette perspective existent. Par contre, il n'existe pas de modèles approchant ces processus d'un point de vue géométrique.

Dans ce mémoire, nous proposons un modèle basé sur les automates cellulaires, capable de simuler les principaux processus qui interviennent dans la formation des brèches. Un automate cellulaire est un modèle discret qui consiste en une grille de cellules pouvant chacune prendre à un instant donné un nombre fini d'états. Le temps est également discret et l'état d'une cellule au temps t est fonction de l'état au temps $t-1$ d'un nombre fini de cellules appelé son voisinage. À chaque nouvelle unité de temps, les mêmes règles sont appliquées pour toutes les cellules de la grille, produisant une nouvelle génération de cellules dépendant entièrement de la génération précédente. Cette approche est compatible avec l'aspect discret de la dissolution des minéraux et permet l'étude de l'évolution géométrique de fragments de roche virtuelle. Plus spécifiquement, on veut mesurer la complexité morphologique des fragments par leur dimension fractale de bordure, une méthode de mesure utilisée sur des échantillons réels et permettant de valider notre modèle avec des données analogiques.

Un simulateur a été conçu pour mettre en oeuvre un tel modèle. Celui-ci est codé en Java et l'interface graphique est en HTML. Des expériences sur le simulateur ont mis en évidence deux régimes de dissolution : l'un limité par la diffusion (*Diffusion Limited Regime* – DLR), l'autre cinétique. Le premier régime dépend de la surface exposée et on y observe l'arrondissement et le lissage progressif des fragments. Le second régime est indépendant de la surface et on observe la formation de cavités dendritiques et une augmentation progressive de la complexité morphologique. D'un point de vue géochimique, le régime DLR est dit «contrôlé par la surface» alors que le régime cinétique est dit «contrôlé par le transport». Les extensions possibles au modèle sont variées et nombreuses.

MOTS CLÉS : Brèche Hydrothermale, Automate Cellulaire, Modélisation, Dissolution, Dimension Fractale

INTRODUCTION

Les brèches hydrothermales sont souvent associées à des types de gisements économiquement intéressants, en particulier pour le zinc, le cuivre, l'or et l'uranium qu'elles contiennent. Comprendre les processus intervenant dans la formation des brèches est donc très important en géologie. Une façon d'arriver à mieux comprendre ces processus consiste à créer des modèles numériques. En général, on utilise des modèles numériques pour confirmer ou réfuter des hypothèses théoriques. Dans le présent cas, nous voulons évaluer la relation entre la surface exposée au fluide d'un fragment et sa morphologie après dissolution.

En fait, la modélisation du processus de dissolution n'est qu'un petite partie d'un grand modèle de bréchification beaucoup plus complexe incluant les processus de fragmentation, de croissance de cristaux, de diffusion-advection, de percolation et de métasomatose. Ces processus géologiques seront expliqués en détail au chapitre 1. Ce modèle devrait être suffisamment générique pour simuler la bréchification des roches de toute nature. Le travail accompli dans le cadre de ce mémoire constitue donc la première étape de conception d'un modèle de brèche virtuelle complet.

L'organisation du mémoire est la suivante. Le premier chapitre présente les fondements théoriques des concepts géologiques sur lesquels se base notre modèle. Le deuxième chapitre est consacré à l'explication des deux paradigmes de modélisation utilisés pour mettre en oeuvre les processus géologiques. Ces deux paradigmes sont la résolution d'équations aux dérivées partielles et les automates cellulaires. Le troisième chapitre explore en détail la mise en oeuvre du simulateur : les structures de données, les principaux modules, les processus et les algorithmes. Le quatrième chapitre présente un guide d'utilisation du simulateur. Le dernier chapitre résume les expériences et les résultats obtenus grâce au simulateur. Soulignons qu'au moment de déposer ce mémoire, un article de revue (32) était en cours d'approbation au *Journal of Structural Geology*.

Chapitre I

NOTIONS DE BASE DE GÉOLOGIE

Nous présentons ici les notions de géologie qui seront nécessaires pour comprendre le problème traité dans ce mémoire. Nous définissons les concepts de brèche hydrothermale, de géométrie fractale ainsi que plusieurs processus intervenant dans la formation et l'altération des brèches.

1.1 Brèches hydrothermales

Une brèche est un ensemble de blocs anguleux noyés dans un ciment — aussi appelé matrice — de nature variable. Les brèches hydrothermales sont caractérisées par un processus de fracturation ainsi qu'une dissolution des fragments causée par des eaux souterraines sous pression à haute température (i.e., plus de 300°C environ).

Contrairement à d'autres types de brèches, les brèches hydrothermales ne font pas intervenir des variations de la pression solide mais bien des variations dans la pression des fluides. Il faut noter que des variations dans la pression des fluides font quand même varier la pression solide. Une diminution de la pression entraîne une bréchification par implosion hydraulique alors qu'une augmentation mène à la formation d'une brèche par explosion (31).

L'étude des brèches, et plus particulièrement celle des brèches hydrothermales, est importante car de nombreux gisements de minéraux sont associés à de vastes ensembles de brèches. De par leurs multiples origines et leur répartition dans des milieux différents, l'étude des brèches est à la rencontre de nombreuses spécialités des géosciences.

Définir les caractérisques des brèches, et en particulier leur géométrie, requiert quatre paramètres : la complexité morphologique des fragments (Dm), la distribution granulométrique (Ds), la fabrique (i.e., l'orientation) et la dilatation (i.e., l'espacement des fragments). La figure 1.1 permet d'observer et de comparer l'effet typique de ces quatre paramètres dans le cas à deux dimensions (30). Soulignons toutefois que les processus réels sont à trois dimensions.

- La complexité morphologique (Dm) indique à quel degré la surface d'un solide est accidentée. Elle peut être analysée soit en utilisant la transformée de Fourier, soit par des méthodes fractales, sujet de la prochaine section. Un Dm élevé indique une complexité morphologique élevée et donc un haut degré de corrosion.
- La granulométrie (Ds) correspond à la distribution des fragments en fonction de leur taille et est caractérisée par des lois complexes. On calcule Ds en faisant le rapport du nombre de petits fragments sur celui de gros fragments. Une explosion est caractérisée par une valeur de Ds élevée et des fragments de tailles variées alors que pour des brèches formées dans des zones contraintes uniformément, les fragments ont approximativement la même taille et la valeur de Ds sera proche de 1.
- La fabrique est l'ensemble des caractères structuraux d'une roche. Elle peut être mesurée grâce à deux paramètres : l'orientation des fragments, exprimée par la moyenne ou le mode[1], et un indice de dispersion tel que l'écart type de la distribution des orientations.
- La dilatation représente l'espacement entre les fragments et peut être calculée par le rapport entre la surface des fragments et la surface de la matrice.

1.2 Géométrie fractale

Depuis sa conception par Benoit Mandelbrot (37), la géométrie fractale a changé la façon de comprendre et d'étudier la nature (6). La géométrie fractale est un langage mathématique pouvant servir à décrire plusieurs phénomènes que les géologues observent. Très tôt dans leur formation de terrain, les géologues apprennent à photographier les roches en plaçant un objet de dimension connue tel qu'un marteau, un pic, un crayon, etc. Pourquoi ? Parce qu'il n'est pas toujours facile de déterminer l'échelle de l'objet photographié (6). En effet, certaines structures ne semblent pas avoir d'échelle caractéristique. On dit de ces objets qu'ils sont autosemblables, c'est-à-dire qu'on peut les observer à différentes échelles et observer presque toujours la même image. Le flocon de Koch montré à la figure 1.2 en est un bon exemple. En sciences de la terre,

[1] Dans le cas d'une variable discrète, le mode est la valeur dont l'effectif (la fréquence) est maximal (43).

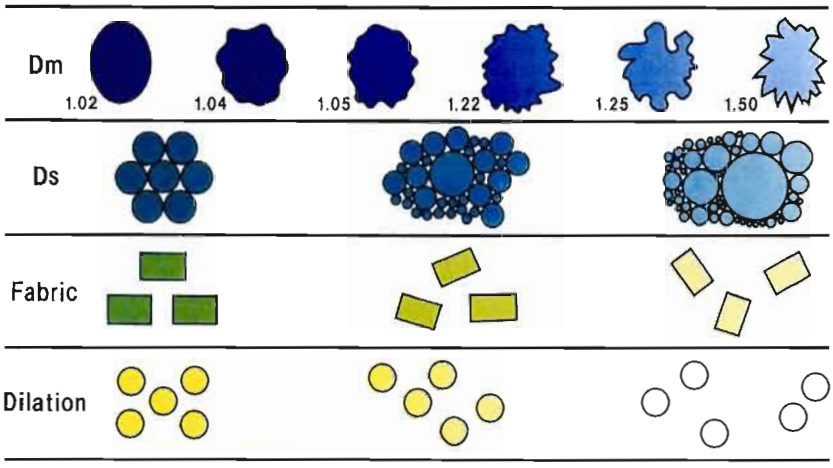


FIG. 1.1 Paramètres géométriques permettant de caractériser une brèche. Note : les termes sont en anglais (30).

des travaux plus avancés sur le sujet ont été faits dans deux domaines : la morphologie des rivières et la fragmentation (31). Dans le cadre de ce travail, notre utilisation de la géométrie fractale sera limitée à la mesure de la complexité morphologique des fragments.

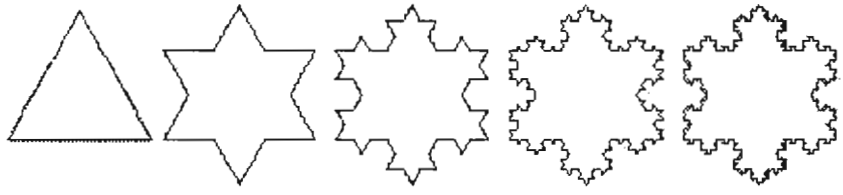


FIG. 1.2 Flocon de Koch, construit par reproduction autosemblable du même motif. Adapté de H. von Koch (56).

Il existe plusieurs méthodes pour déterminer la dimension fractale d'un objet, méthodes qui dépendent du type d'objet que l'on veut étudier. Les deux plus connues sont la méthode du diviseur et la méthode des boîtes. C'est cette dernière qui retiendra notre attention. Il nous faut tout d'abord une image en noir et blanc. Le noir représente l'objet dont on veut mesurer la dimension fractale. On divise ensuite l'image en morceaux de tailles identiques en lui superposant une grille pour laquelle chaque carré portera le nom de *boîte* (voir figure 1.3). Puis, on compte le nombre de boîtes contenant du noir et on déduit une probabilité de présence. On recommence

l'opération mais en faisant varier la taille des boîtes. On place ensuite les deux probabilités de présence sur un plan cartésien et on tire une droite reliant chaque point. La pente de cette droite correspond à la dimension fractale. Pour une plus grande précision, on peut faire varier plusieurs fois la taille des boîtes. Ce calcul est très facile à effectuer sur un ordinateur avec le logiciel approprié. Pour mesurer des fragments d'une brèche, il suffit d'en prendre une photo numérisée et d'en faire l'édition de manière à garder seulement l'intérieur en noir et l'extérieur en blanc. Le logiciel peut alors utiliser cette image et effectuer le travail décrit ci-haut. C'est la méthode utilisée pour calculer la dimension fractale de fragments de roche réels. La dimension fractale permet de mesurer et de comparer la complexité morphologique de fragments de brèches.

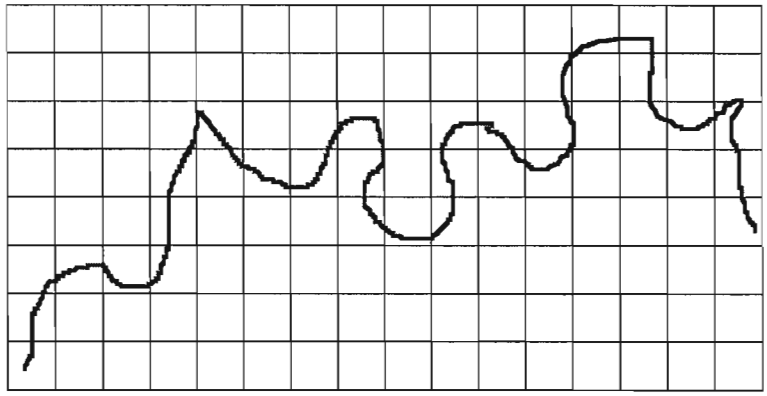


FIG. 1.3 Grille pour la méthode des boîtes : on compte le nombre de carrés touchés par le tracé.

1.3 Mécanismes de formation et d'altération des brèches

1.3.1 Fragmentation de la roche

La fragmentation de la roche, c'est l'action de briser, fracturer ou morceler la roche en plusieurs fragments. Il existe deux grandes géométries (ou catégories) de fragmentation en géologie : la fragmentation régulière et la fragmentation fractale. De manière générale, la propagation des fractures sera lente dans les systèmes de fracturation nécessitant une faible intensité énergétique. Les fragments résultants seront de tailles approximativement identiques, ce qui conduira à une distribution de type gaussienne (39). La propagation des fractures sera rapide pour les systèmes de fracturation nécessitant une forte intensité énergétique, ce qui mènera à une distribution des fractures proche d'une loi log-normale. Une analyse fractale du milieu résultant permet d'observer une distribution fractale de la taille des fragments avec un Ds proportionnel

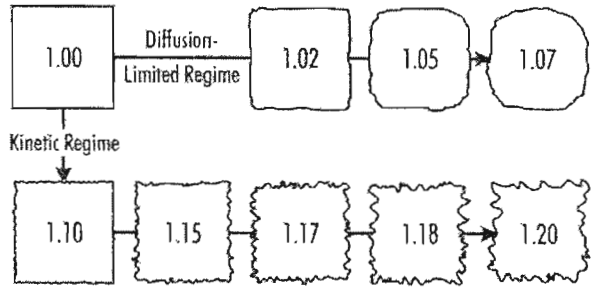


FIG. 1.4 Comparaison des régimes cinétique et DLR. Les valeurs numériques représentent le Dm caractérisant une brèche (voir la section 1.1).

à l'intensité énergétique (53). On observera ainsi un petit nombre de gros fragments et un très grand nombre de petits fragments (voir figure 1.1). À noter qu'il est difficile de trouver des lois simples reliant l'énergie de la bréchification et la morphologie du milieu résultant (30).

1.3.2 Dissolution

La dissolution est l'action de dissoudre une substance dans une autre. Il s'agit d'un processus important agissant au sein des brèches. La vitesse de dissolution d'une roche est influencée par la température, la pression, la nature et le pH du fluide, la nature des minéraux, le blindage ou l'enduit (13), la surface ainsi que la perméabilité.

Deux types de régimes de dissolution ont été mis en évidence par Sahimi et Tsotsis (48). Le premier, dit «cinétique», est caractérisé par un taux de dissolution limité uniquement par la vitesse de réaction chimique qui corrode la surface d'un fragment indépendamment de la surface exposée et qui laisse le fragment dans un état hautement accidenté. L'autre mécanisme est dit «limité par la diffusion» (*Diffusion Limited Regime* — DLR) et seules les parties les plus exposées du solide sont atteintes par le fluide. Le taux de dissolution du régime DLR est donc dépendant de la surface exposée au fluide et est caractérisé par l'arrondissement et le lissage du solide (29). La figure 1.4 montre les deux régimes et leurs effets sur un fragment.

Il est possible d'analyser la morphologie des fragments corrodés en mesurant leur dimension fractale. Cette mesure exprime le degré d'irrégularité (complexité) d'un objet. Il existe d'autres méthodes pour analyser la morphologie des fragments d'une brèche. Par exemple, lorsque l'on connaît la taille et la forme de l'objet initial, on peut utiliser le rapport Sur-

face/Volume pour mesurer l'évolution de la complexité de l'objet (3). En effet, plus la surface d'un objet est corrodée et complexe, plus le rapport Surface/Volume sera élevé. Cette méthode a l'inconvénient majeur d'être utilisable uniquement en simulation car il est expérimentalement très ardu, voire impossible, de mesurer avec une grande exactitude la surface d'un fragment réel. La méthode ne permet donc pas facilement la comparaison entre des valeurs expérimentales et des résultats de simulation.

Le mécanisme de dissolution cinétique peut mener à l'apparition de nouveaux fragments à mesure que le morceau se corrode. Ce faisant, la dimension fractale diminue comme on peut le voir sur le dernier fragment de la figure 1.5 (19).

La dissolution s'accompagne souvent de métasomatose (voir la section 1.3.5) sur les parois et, dans une certaine mesure selon la pénétration interne du fluide, à l'intérieur des fragments. Ce changement de composition externe du solide peut mener au blindage du fragment, c'est-à-dire que le processus de dissolution sera ralenti voire même bloqué à cause des propriétés chimiques du nouveau minéral (32; 13).

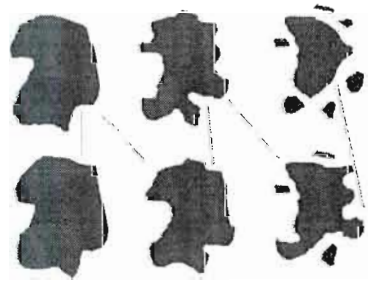


FIG. 1.5 Création de nouveaux fragments par le processus de dissolution et diminution de la dimension fractale : les dimensions fractales sont respectivement : 1.017, 1.046, 1.046, 1.052, 1.053 et 1.016 (commençant en bas à gauche et terminant en haut à droite).

1.3.3 Précipitation

Le mécanisme de précipitation (i.e., croissance des cristaux) est un processus antagoniste à celui de la dissolution. Les systèmes solide–fluide tendent généralement vers un équilibre local entre la dissolution et la précipitation. On distingue deux facteurs qui influencent ou contrôlent la croissance des cristaux : la surface et le transport. Il semble que le taux de croissance des cristaux soit limité par le facteur le plus lent des deux. On dira que le processus de croissance

est *limité par la surface* quand l'arrivée des éléments est rapide et que le processus d'accrochage est lent. Quand le processus d'accrochage des éléments est rapide et que l'arrivée d'éléments est lente, on dira que le processus est *limité par le transport*. D'autres facteurs influencent la vitesse des processus et déterminent quel sera le facteur de croissance limitant. La température, la pression, le taux de sursaturation du fluide et la morphologie de la surface en sont d'autres. En général, les fragments à la morphologie lisse sont associés à un fluide peu sursaturé, donc à une croissance lente des cristaux, la surface étant alors le facteur limitant. Les fragments de morphologie complexe sont plutôt associés à un fluide hautement sursaturé, à une croissance rapide des cristaux et, donc, à un processus limité par le transport. Les morphologies complexes sont souvent associées à des textures dites dendritiques[2] (exemple : un flocon de neige). La surface lisse des systèmes à croissance lente est liée à un rapport des vitesses de dissolution/précipitation se rapprochant de 1. Les excroissances dendritiques sont vraisemblablement dissoutes rapidement car leur liaison avec les autres cristaux est faible.

1.3.4 Percolation

La percolation est souvent définie comme le mouvement de l'eau dans un milieu poreux saturé (43). On peut la modéliser en se basant sur la loi de Darcy. Cette loi décrit la vitesse d'écoulement de l'eau dans un corps poreux, et fut formulée en 1856 par Henri Darcy à la suite de travaux approfondis sur l'écoulement de l'eau dans une couche filtrante de sable (50). La vitesse d'écoulement de l'eau dans un milieu poreux dépend de la perméabilité du milieu. La perméabilité est une des variables les plus importantes déterminant la nature des écoulements de fluide. C'est également une des plus difficile à prédire. Les facteurs les plus important semblant influencer la perméabilité seraient la topologie, la géométrie du réseau formé par les pores connectés (35) et la taille des grains. La figure 1.6 montre le rendu[3] d'une définition de milieu poreux avec une porosité de 7.5% (38).

---

[2]Une dendrite est une structure qui évolue ou croît vers une forme d'arbre, c'est-à-dire se séparant en plusieurs branches. Les dentrites cristallines forment naturellement des structures fractales.

[3]Représentation d'un modèle 3D par l'affichage et le traitement de ses surfaces à partir de paramètres de texture, de couleur, d'éclairage et d'ombrage (43) .

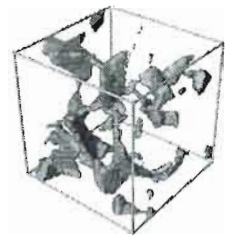


FIG. 1.6 Réseau de pores.

Une sous-catégorie de percolation bien connue est la percolation envahissante (*invasion percolation*) qu'on retrouve dans les systèmes de déplacement fluide–fluide (eau–pétrole, eau–glycérol, etc.) en milieu poreux (58). Dans ces systèmes, les forces capillaires dominent les forces de viscosité. Plus un pore est étroit, plus la force capillaire domine (6). Dans les milieux de porosité ordonnée (porosité aléatoirement répartie), la croissance des structures envahissantes donne lieu à des formes régulières alors que dans les milieux désordonnés, les formes observées ont une géométrie fractale. La figure 1.7 est un exemple de percolation envahissante (16). On peut observer que le fluide envahissant (noir) évolue vers la droite et emprisonne parfois des régions de fluide défendant[4] (blanc).

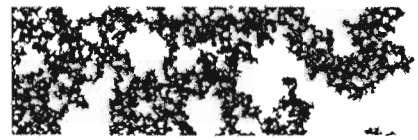


FIG. 1.7 Un exemple de percolation envahissante.

La percolation joue également un rôle important dans l'apport et le départ d'éléments. Ce phénomène sera expliqué un peu plus en détail dans la prochaine section.

1.3.5 Diffusion

Le processus de diffusion dans la roche a une grande importance dans l'évolution des brèches hydrothermales. La diffusion correspond à un transfert d'éléments sans déplacement de

[4]Par opposition à envahissant.

fluide. C'est par ce processus que les ions se propagent dans la roche. On distingue deux types de diffusion : la diffusion dans les solides et la diffusion dans les fluides. Ce processus extrêmement lent est souvent négligé dans les modèles numériques (33). En effet, les coefficients de diffusion sont de l'ordre de 10^{-10} à 10^{-14} cm^2/s pour des températures de 800° C. Ceci veut dire qu'après 1 à 10 millions d'années, les distances de diffusion ne devraient pas avoir dépassées 10 cm (17).

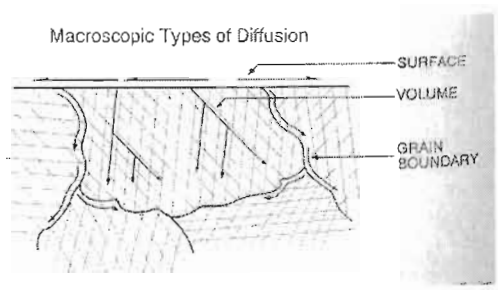


FIG. 1.8 Les trois types de diffusion macroscopique : surface, volume et joints (35).

La diffusion correspond à la loi de Fick. Lorsqu'un solide contient de l'eau, on observe des coefficients de diffusion beaucoup plus élevés. Ceux-ci sont de l'ordre de 10^{-5} cm^2/s. Les ions sont diffusés dans le fluide adsorbé[5] sur la surface, sur les joints et à l'intérieur des grains[6]. La figure 1.8 montre ces trois mécanismes de diffusion. La diffusion de surface influence surtout la formation de cristaux (voir section 1.3.3). La différence de vitesse entre la diffusion de surface et la diffusion sur les joints est habituellement négligeable (35). Par contre, on remarque que la vitesse de diffusion sur les joints peut être de l'ordre de 10^4 fois supérieure à celle de la diffusion de volume pour des températures de quelques centaines de degrés. À très haute température, la diffusion à l'intérieur des grains devient dominante à cause du plus grand nombre de canaux de diffusion (35; 30). La nature du minéral et du fluide jouent également un rôle dans la vitesse de diffusion.

La diffusion joue un rôle important dans le transport d'éléments chimiques et en particulier pour la métasomatose. Bernard Guy (24) définit la métasomatose comme suit :

Transformation non isochimique, c'est-à-dire transformation mettant en jeu des ap-

[5] Adsorption : adhésion physique ou physico-chimique à la surface d'un corps de substances en solution ou en suspension dans un fluide (43).

[6] Aussi appelé diffusion de volume.

ports et départs d'éléments chimiques et modifiant donc la composition chimique de la roche de départ.

La théorie élaborée par Korzhinskii (34) est celle généralement acceptée dans la littérature scientifique pour expliquer la formation des zones métasomatiques. Les zones d'échanges métasomatiques sont caractérisées par l'apparition de discontinuités spatiales de compositions ou de concentrations (fronts) (24).

FIG. 1.9 Exemple de métasomatose le long d'une fracture : on peut observer les différentes couleurs de minéraux associées à différentes phases.

Partout où il y a mouvement du fluide par rapport à un solide avec une tendance à atteindre un équilibre chimique local, on peut observer des changements chimiques radicaux même si les conditions initiales sont continues. L'apparition de fronts est d'abord et avant tout un problème d'échelle (24). Lorsqu'on fait le rapport entre l'équation de diffusion $x = (Dt)1/2$ et l'équation de percolation $x = vt$ (où D est le coefficient de diffusion, t est le temps, v est la vélocité et x est l'espace), le rapport devient 0 lorsque t tend vers l'infini. Lorsque t et/ou x deviennent grands, les distances induites par la diffusion deviennent négligeables. Cela est toujours vrai, peu importe la valeur de v et de D, ce qui signifie que pour un certain point de vue et pour une certaine échelle, la diffusion n'est pas négligeable et le front ne ressemble plus à une discontinuité. On dirait plutôt que le front s'étale. Pour bien comprendre ce phénomène, imaginons que l'on trempe une poche de thé dans de l'eau. Si on laisse les éléments diffuser dans l'eau suffisamment longtemps sans introduire de mouvement, on voit un halo de particules de thé se former autour de la poche. Le halo semble se terminer de manière abrupte, claire, précise. Or, cela dépend de l'échelle à laquelle nous regardons. Avec l'aide d'une loupe de force suffisante, on s'apercevrait qu'en fait, le front, la discontinuité observée, s'étale. L'apparition de fronts métasomatiques est un phénomène fréquemment observable sur les fragments des brèches

hydrothermales. La figure 1.10 illustre les explications de ce paragraphe.

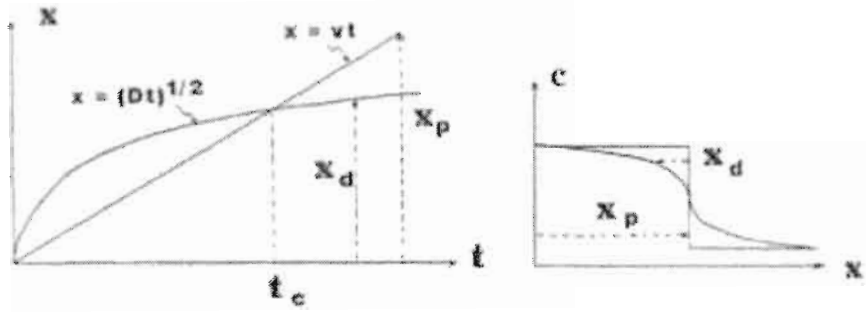


FIG. 1.10 Le diagramme de gauche montre les équations x = (Dt)1/2 et x = vt dont il est question dans le texte. On voit que lorsque t tend vers l'infini, les distances (x) de diffusion deviennent négligeables par rapport aux distances de percolation. La figure de droite permet de comparer l'étalement des concentrations par rapport à l'espace lorsque l'échelle est suffisante (24).

Chapitre II

MODÉLISATION NUMÉRIQUE DE PROCESSUS GÉOLOGIQUES

La modélisation numérique, aussi appelée simulation par ordinateur, permet d'illustrer les comportements de systèmes complexes, habituellement basés sur des équations mathématiques. La modélisation numérique est utilisée par les scientifiques et les ingénieurs pour confirmer ou réfuter leurs hypothèses théoriques. Certaines techniques de programmation sont utilisées encore et encore en modélisation numérique : le calcul de mouvement des particules, la résolution de matrices et le calcul en grille (2). C'est sur cette dernière technique que nous nous concentrerons dans ce chapitre.

Le calcul en grille regroupe les problèmes pouvant être modélisés par des équations aux dérivées partielles (par exemple, les flux de chaleur, la diffusion de l'eau dans la roche, les turbulences d'un fluide, le passage de l'air sur une aile d'avion, etc.) ainsi que par des équations différentielles ordinaires. Les automates cellulaires et certains problèmes de traitement d'images peuvent également être considérés comme du calcul en grille. On peut même résoudre certaines catégories de problèmes de matrices grâce à cette technique (matrices strictement diagonales dominantes). Habituellement, de tels problèmes sont résolus en approximant la solution à l'aide d'une méthode itérative. La méthode itérative la plus élémentaire est l'itération de Jacobi. Il existe plusieurs autres méthodes itératives plus efficaces : Gauss-Seidel, *successive over-relaxation* (SOR) et *multigrid*, pour n'en nommer que quelques-unes (25; 2).

2.1 Équations aux dérivées partielles

Les équations différentielles aux dérivées partielles — EDP — sont importantes dans plusieurs domaines de la physique pour décrire les phénomènes qui varient de manière continue dans le temps et l'espace. On s'en sert notamment pour les problèmes de diffusion et de propagation,

en mécanique des fluides, en mécanique quantique et dans plusieurs autres domaines. Il n'existe aucune méthode universelle pour les résoudre toutes. La plupart des EDP n'ont pas de solution analytique. C'est pourquoi il faut souvent utiliser des méthodes numériques pour les résoudre. Ce sont les méthodes de résolution numérique qui nous intéressent dans ce cas-ci.

Les définitions suivantes proviennent des notes du cours MATH-303 du Cégep de Maisonneuve (27).

Définition 1 (Équation différentielle) *Équation contenant des dérivées.*

Définition 2 (Équation différentielle ordinaire (EDO)) *Équation différentielle ne possédant qu'une seule variable indépendante.*

Définition 3 (Équation différentielle aux dérivées partielles (EDP)) *Équation différentielle possédant plus d'une variable indépendante en plus de la variable dépendante.*

Définition 4 (Ordre) *L'ordre d'une équation différentielle est celui de la dérivée de l'ordre le plus élevé qui y figure. Par exemple, si une équation contient des dérivées premières et des dérivées secondes, l'ordre le plus élevé est 2.*

Définition 5 (Solution générale) *La solution générale d'une équation différentielle d'ordre n est une solution qui contient n constantes arbitraires et indépendantes. La solution générale à une EDO consiste en une famille de courbes qui satisfont l'équation différentielle donnée. Chaque courbe représente une solution particulière.*

Définition 6 (Solution particulière) *La solution particulière d'une équation différentielle d'ordre n est une solution dans laquelle les constantes sont déterminées, généralement à l'aide d'une ou plusieurs hypothèses sur la variable dépendante, hypothèses appelées* conditions initiales.

2.1.1 Types d'équations

Soit une EDP d'ordre 2 de la forme suivante[1] :

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu = G \tag{2.1}$$

[1] L'équation de la diffusion décrite à la section 2.1.3 est une équation d'ordre 2.

Une telle équation est dite *hyperbolique* au point (x_0, y_0) si et seulement si ce point est élément du domaine D de la solution et $B(x_0, y_0)^2 - 4A(x_0, y_0)C(x_0, y_0)$ est positif. Si l'équation est hyperbolique pour tous les points du domaine D, on dit alors que l'équation est hyperbolique. Si $B(x_0, y_0)^2 - 4A(x_0, y_0)C(x_0, y_0)$ est nul pour tous les points de D, alors l'équation est dite *parabolique*. Si c'est négatif, alors l'équation est dite *elliptique* (7; 46).

En physique, il existe des équations d'importance pour chaque type. Par exemple, l'équation de Laplace et l'équation de Poisson sont des équations dites elliptiques. L'équation de la diffusion est un excellent exemple d'équation parabolique. Les équations paraboliques sont généralement faciles à résoudre de manière numérique. L'équation d'onde est un bon exemple d'équation hyperbolique. Les équations hyperboliques sont généralement plus complexe à résoudre numériquement que les équations paraboliques.

2.1.2 Méthodes de résolution numérique

En général, les équations différentielles ordinaires — EDO — sont plus faciles à résoudre analytiquement que les équations différentielles partielles — EDP. Plusieurs méthodes analytiques existent pour résoudre les EDP : la transformée de Bäcklund, les équations caractéristiques, la fonction de Green, la transformée en intégrale, la paire Lax et la séparation des variables en sont quelques exemples (46). Il existe également un grand nombre de méthodes pour résoudre les EDO.

De manière générale, toutes les méthodes numériques utilisent ce qu'on appelle une méthode itérative. Il existe plusieurs méthodes itératives, les principales étant Jacobi, Gauss-Seidel, *successive over-relaxation* (SOR) et *multigrid* (2). Les méthodes Jacobi et Gauss-Seidel et SOR sont faciles à implémenter ; par contre, les méthodes de type *multigrid* sont plus complexes. Dans un contexte où plusieurs modèles interagissent ensemble, SOR est particulièrement intéressante car elle possède un paramètre *omega* permettant de contrôler la vitesse de convergence.

Un exemple de méthode d'analyse numérique est appelée la méthode par éléments finis (*Finite Element Method* — FEM). Il existe de nombreux travaux sur le sujet (51). La méthode par éléments finis est particulièrement utile lorsqu'on veut résoudre par approximation des équations différentielles partielles et que certains paramètres sont variables (par exemple, le coefficient de diffusion qui varierait en fonction de la nature du minéral). Les fondements mathématiques solides et la grande généralité de cette méthode font qu'elle est adaptable à un grand nombre de

problèmes. La méthode Crank-Nicholson (12) est un cas particulier de la méthode par éléments finis. Par une série de transformation mathématiques, celle-ci ramène l'équation différentielle partielle à une équation différentielle ordinaire. Il faut alors utiliser la méthode par différence finie (*Finite Difference Method*). Cette dernière méthode consiste à solutionner l'équation par approximation en calculant un nombre fini de points et en utilisant une méthode itérative. Le plus grand défi consiste à trouver une équation qui soit une bonne approximation de l'équation étudiée et qui soit numériquement stable. Par numériquement stable, on entend que l'erreur combinée des données en entrée et des calculs intermédiaires ne s'accumulent pas de manière à rendre le résultat final inutilisable.

2.1.3 Équation de la diffusion

La technique de modélisation du processus de diffusion est basée sur la résolution d'une équation différentielle partielle de type *parabolique*. Le choix de l'équation exacte dépend si le système est ou non à l'équilibre. Lorsque la quantité de matière reste constante pour une surface ou un volume, et que le flux de diffusion est le même dans toutes les directions et ne change pas avec le temps, on dira que le système a atteint l'équilibre (*steady state*). Cet état est décrit par la première loi de Fick.

$$J = -D\frac{dC}{dx} \tag{2.2}$$

Si les conditions nous montrent que l'état d'équilibre n'est pas atteint (*non-steady state* ou *transient state*), il n'est plus possible d'utiliser la première loi de Fick et les concentrations changent avec le temps. L'équation correspondant à cet état s'appelle la seconde loi de Fick.

$$\frac{\partial C}{\partial t} = -D\frac{\partial^2 C}{\partial x^2} \tag{2.3}$$

Toutefois, ces deux équations ne s'appliquent que rarement à des problèmes réels puisqu'elles sont en seule une dimension. La seconde loi de Fick en deux ou en trois dimensions s'énonce respectivement comme suit :

$$\frac{\partial C}{\partial t} = D(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2}) \tag{2.4}$$

$$\frac{\partial C}{\partial t} = D(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2} + \frac{\partial^2 C}{\partial z^2}) \tag{2.5}$$

Finalement, l'équation de diffusion fait partie d'une classe d'équations comprenant également la loi de Darcy en hydraulique, la loi de Ohm en électricité et la loi de Fourier en thermodynamique. Le tableau 2.1 résume bien où se situe la loi de Fick au sein de cette grande classe d'équations. On comprend donc que l'expression «équation de la diffusion» est une simplification. Pour être tout à fait rigoureux, il aurait donc fallut préciser que nous travaillons avec l'équation de la diffusion des solutés, qui correspond à la loi de Fick.

Gradient / Flux	Pression ou charge hydraulique	Potentiel électrique	Température	Concentration
Fluide	**DARCY**	Électro-osmose		Osmose
Électricité	Rouss	**OHM**	Seebek ou Thompson	
Chaleur		Peltier	**FOURIER**	Dufour
Soluté			Soret	**FICK**

FIG. 2.1 L'équation de la diffusion : un gradient qui produit un flux. En caractère gras sur la diagonale, l'équation de la diffusion porte alors le nom loi de Darcy, loi de Ohm, etc. (Adapté de Alain Rouleau, UQAC (45)).

2.2 Automates cellulaires

2.2.1 Historique

Pendant longtemps, les scientifiques croyaient que la meilleure façon de modéliser la nature était à l'aide d'équations mathématiques. Au début des années 80, les travaux de Stephen Wolfram (59) ont montré qu'il était possible de reproduire sur ordinateur des phénomènes observables dans la nature à l'aide de règles simples. Historiquement, le concept des automates cellulaires — AC — a véritablement fait son apparition vers la fin des années 40 avec les travaux sur l'auto-reproduction de John von Neumann (57) et les expérimentations de Stanislaw Ulam (54). Toutefois, il s'est écoulé plusieurs années avant que renaisse un intérêt pour les AC. Au début des années 70, Martin Gardner publie un article dans la revue *Scientific American* sur le «Jeu de la Vie» (20), inventé par John Conway. Le jeu en question possède aujourd'hui

une notoriété quasi-universelle dans les milieux scientifiques. Il permet de générer des comportements extrêmement complexes à partir de quelques règles simples. À partir de ce moment, de nombreux scientifiques amateurs se sont intéressés aux AC et tentèrent de trouver d'autres règles simples pouvant générer des comportements intéressants. Curieusement, très peu de travaux scientifiques furent entrepris ou publiés suite à la publication de Martin Gardner. C'est vraiment depuis les travaux de Stephen Wolfram qu'on observe un regain d'intérêt pour les AC chez les scientifiques.

2.2.2 Définitions

Un automate cellulaire peut être vu comme un modèle où l'espace, le temps et les grandeurs physiques prennent des valeurs discrètes. L'espace est représenté par une matrice de cellules, avec une, deux, ou plusieurs dimensions[2], selon le problème à traiter. Chaque cellule peut, à un instant donné, être dans un nombre fini d'états. Ces états sont habituellement associés et représentés graphiquement par des couleurs. L'état d'une cellule au temps t est fonction de l'état, au temps $t-1$, d'un nombre fini de cellules appelé son voisinage. À chaque nouvelle unité de temps, les mêmes règles sont appliquées pour toutes les cellules de la grille, produisant une nouvelle génération de cellules dépendant entièrement de la génération précédente.

De tels systèmes sont appelés automates cellulaires à la condition que ceux-ci rencontrent les trois propriétés fondamentales suivantes (47) :

1. Le parallélisme : L'ensemble des constituants de l'automate évoluent simultanément.
2. La localité : Le nouvel état d'une cellule ne dépend que de son état actuel et de l'état des cellules de son voisinage immédiat.
3. L'homogénéité : Les lois sont universelles, c'est-à-dire communes à l'ensemble de l'espace de l'automate cellulaire.

Les automates cellulaires peuvent servir à simuler des systèmes physiques continus décrits par des équations différentielles partielles. Contrairement aux équations différentielles, l'idée générale d'un automate cellulaire n'est pas de décrire analytiquement un système complexe à l'aide de longues équations (approche descendante) mais bien de laisser la complexité émerger par l'interaction d'éléments simples, suivant des règles simples (approche ascendante).

---

[2]Toutefois, au-delà de quatre dimensions, l'interprétation devient quelque peu difficile.

2.2.3 Classification et propriétés

Il faut tout d'abord préciser qu'il existe plusieurs modèles de classification des AC. Celui de Wolfram, un modèle comportant quatre classes possibles, fut le premier proposé et est aujourd'hui le plus connu (60; 41). La classe I regroupe les systèmes qui se terminent rapidement en un état fixe. Les systèmes de la classe II développent des comportements qui se répètent périodiquement. Ceux de la classe III sont dit chaotiques parce qu'ils évoluent continuellement de manière imprévisible et aléatoire. Finalement, les systèmes de la classe IV sont dit structurés parce qu'ils peuvent développer des structures complexes bien que parfois instables. Il faut noter que cette classification n'est pas universellement acceptée par les chercheurs de ce domaine. En effet, il n'existe pas d'algorithme permettant de déterminer clairement la classe d'un AC. David Eppstein, qui s'intéresse aux phénomènes des planeurs[3], a mis en évidence le «problème de la classe IV» du modèle de Wolfram (15). Ainsi, il n'existe pas de test efficace pour déterminer si une règle mène à un AC de la classe IV. Eppstein propose une nouvelle classification plus simple. Mais de façon générale, le problème du classement des automates cellulaires reste ouvert. Il n'existe toujours pas de façon de classer un AC *a priori*, c'est-à-dire, à partir de ses règles de transitions. L'intérêt de connaître la classe d'un AC est heureusement peu important en pratique.

AC réversibles et AC aléatoires

On dit d'un automate cellulaire qu'il est réversible s'il ne se produit pas de perte d'information au cours de l'évolution de celui-ci (4). C'est le cas lorsque chaque configuration possède un unique successeur et un unique prédécesseur. Un AC réversible est donc déterministe : à partir d'une même configuration initiale, l'automate évolue toujours de la même façon. Certains AC évoluent selon une fonction aléatoire. Ces automates sont dit probabilistes. De tels automates sont donc toujours non réversibles. De plus, chaque simulation est unique à cause de la nature non déterministe de ce type d'AC.

---

[3]Les planeurs, de l'anglais *glider*, sont des structures qui se déplacent d'un bout à l'autre de la grille de l'AC et qui réapparaissent après un certain nombre de générations avec la même orientation mais à un endroit différent.

2.2.4 Conception

D'une manière générale, on peut construire tout type d'automate cellulaire en jouant sur les règles structurelles et fonctionnelles. Les premières définissent la structure spatiale du réseau d'automates, soit sa géométrie. Ces règles structurelles sont le nombre de dimensions, le mode d'arrangement des cellules (i.e., carré, triangulaire ou hexagonal pour une matrice en deux dimensions) et la détermination du voisinage. Les secondes déterminent le nombre d'états et les règles de transition. Le choix de ces deux types de règles permet de construire un univers (modèle) adapté à l'objectif recherché.

Matrice

Certains auteurs emploient les termes *grille*, *treillis*, *réseau*, voire même *échiquier*, pour désigner l'espace contenant les cellules sur un automate cellulaire. Tous ces termes sous-entendent que l'AC est bidimensionnel. Or, comme nous travaillerons sur des espaces à deux et trois dimensions, nous utiliserons plutôt le terme *matrice* pour désigner l'espace d'un AC.

La matrice, que celle-ci soit bidimensionnelle ou tridimensionnelle, n'est pas limitée à des cellules carrées ou cubiques. En fait, certains phénomènes sont plus facilement modélisables lorsque la géométrie des cellules est différente. Les géométries triangulaires et hexagonales sont celles les plus couramment utilisées. Les figure 2.2 et 2.3 montrent, respectivement, diverses géométries en 2D et en 3D.

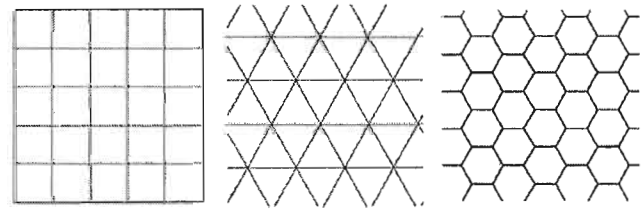


FIG. 2.2 Les géométries les plus populaires pour les matrices bidimensionnelles : carrés, triangulaires et hexagonales (tiré de MathWorld (46)).

Voisinage

Soit une matrice M bidimensionnelle, le voisinage est l'ensemble des cellules entourant la cellule $M(i,j)$ qui peuvent avoir un impact sur celle-ci. La cellule $M(i,j)$ ne fait normalement

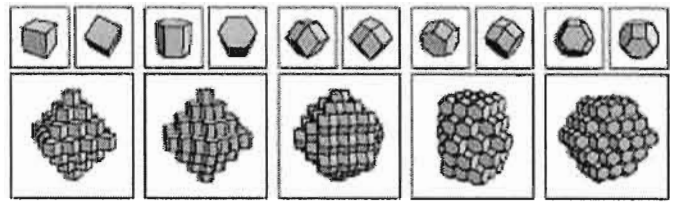


FIG. 2.3 Différents types de polyèdres pouvant servir de cellules pour une matrice tridimensionnelle (source inconnue).

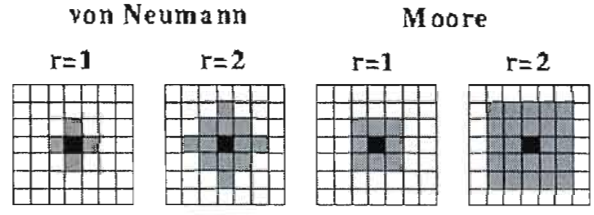


FIG. 2.4 Voisinages les plus couramment utilisés (adapté de Alesdar.org (8)).

pas partie du voisinage. Les voisinages les plus connus sont ceux de von Neumann, Moore et Margolus[4]. On peut également utiliser d'autres types de voisinages mais leur application à des modèles du monde réel semble restreinte. Comme la Figure 2.4 le démontre, les voisinages ne sont pas restreint à un rayon de 1.

Conditions de bordure

Même si en théorie un automate cellulaire ne devrait pas avoir de borne, la mise en oeuvre sur un ordinateur requiert de définir des matrices de taille finie. On définira alors des conditions aux bordures qui permettront de simuler une matrice de taille infinie (par exemple, en changeant les valeurs des bordures de manière périodique). Toutefois, il arrive souvent que le phénomène naturel qu'on désire simuler soit défini pour un environnement bien précis et nécessite des contraintes pour être réaliste. Dans pareil cas, on utilise une condition de bordure à valeur fixe qui consiste à définir des bornes et à les garder tout au long de l'exécution de l'AC. C'est ce type de condition de bordure qui sera utilisé dans le simulateur.

---

[4]Le voisinage de Margolus, défini sur des blocs de 2x2 cellules sur des matrices en deux dimensions, est utile pour la modélisation de systèmes physiques (par exemple : le modèle *lattice-gaz*).

Conditions initiales

Les conditions initiales peuvent être générées aléatoirement ou être spécifiées de façon particulière. Une façon courante de générer aléatoirement une condition initiale est de définir un nombre de cellules proportionnel à un pourcentage prédéfini auquel correspond un état possible. Par exemple, si les états possibles sont un nombre fini de couleurs (bleu, rouge et jaune), alors on fera correspondre à chaque état un pourcentage pertinent pour le problème :

- Bleu : 10%
- Rouge : 20%
- Jaune : 70%

Par conséquent, si le nombre total de cellules de la matrice est 100, on distribuera de façon aléatoire ou particulière (selon le problème), 10 cellules à l'état bleu, 20 cellules à l'état rouge et 70 cellules à l'état jaune.

Règles de transition

Les règles de transition définissent la nature des interactions entre les cellules. Les règles de transition sont par définition homogènes et appliquées de façon simultanée sur l'ensemble des cellules de la matrice. Les règles ne sont ni plus ni moins qu'un algorithme qui permet de déterminer vers quel état évoluera une cellule à la prochaine itération en fonction de l'état des cellules de son voisinage.

Peu de choses sont connues sur la conception des règles de transition. Ainsi, il n'existe aucun algorithme permettant de déduire systématiquement les règles à utiliser pour produire un certain comportement observable ou mesurable des AC. L'ensemble des règles de transition que peuvent prendre un automate est infini. Une règle traditionnelle consiste à compter le nombre de cellules du voisinage qui sont à l'état E_0, E_1, E_2, ..., E_n, et à décider de manière prédéterminée (20) ou de manière probabiliste (32) l'état suivant.

2.2.5 Utilisation des automates cellulaires en modélisation

De nombreux modèles sont basés sur les AC. Il serait trop long de les décrire tous en détail mais voici une liste, non exhaustive, de phénomènes ayant été modélisés par des AC :

- Feux de forêts (5)

- Traffic automobile dans un réseau de rues (10)
- Étalement urbain (55)
- Dissolution de cristaux (32), fonte de glace (52)
- Croissance des arbres, des feuilles, des fleurs, des plantes, des mollusques (61), des coraux (33), des cristaux (flocons de neige) (1)
- Turbulence dans les fluides, dynamique des fluides, dynamique des gaz (i.e., modèles lattice-gaz) (18)
- La diffusion (thermique, électrique, chimique, etc) (25)

2.3 Conclusion

Les deux paradigmes de modélisation que nous avons introduits dans ce chapitre sont applicables dans des situations variées. Dans le prochain chapitre, nous verrons comment chaque processus géologique décrit au chapitre 1 peut être modélisé à l'aide de l'une des deux approches. Le choix de l'approche variera selon la situation.

Chapitre III

CONCEPTION ET MISE EN OEUVRE DU SIMULATEUR DE BRÈCHES HYDROTHERMALES

Le simulateur de brèches hydrothermales modélise plusieurs processus géologiques parmi ceux vus au chapitre 1. Bien que le simulateur pourrait intégrer tous ces processus, pour l'instant, il permet seulement de simuler la dissolution, la précipitation, la diffusion et les échanges métasomatiques. Les processus de dissolution, de précipitation et d'échanges métasomatiques sont modélisés par des automates cellulaires alors que le processus de diffusion est modélisé par une équation aux dérivées partielles. Ce chapitre présente en détail la mise en oeuvre du simulateur de brèches hydrothermales.

3.1 Conception

3.1.1 Structures de données

La pierre angulaire du simulateur est la roche virtuelle, une matrice de deux ou trois dimensions, de taille paramétrable. Chaque cellule de la matrice contient plusieurs champs utilisés par les différents modèles géologiques. Les trois automates cellulaires utilisent le même champ, nommé `nature` (i.e., nature du minéral), pour décrire l'état de l'automate. L'algorithme de diffusion et l'automate cellulaire d'échanges métasomatiques utilisent le même champ appelé `concentration` (i.e., matière en solution). L'algorithme d'identification de nouveaux fragments nécessite un champ `idFragment` pour associer à chaque cellule un identifiant unique.

En résumé, chaque cellule de la matrice contient trois champs : `nature` (`int`), `concentration` (`double`) et `idFragment` (`int`). Sur la plupart des ordinateurs modernes, un `int` occupe 32 bits et un `double` occupe 64 bits. Chaque cellule totalise 128 bits d'information, donc 16 octets.

Dans le but de donner un ordre de grandeur de la quantité d'espace mémoire nécessaire, voici quelques exemples de simulation possibles :

- Une simulation en deux dimensions utilisant une matrice carré de 1000 par 1000 cellules nécessiterait 1 million de cellules de 16 octets, donc 16 Mo.
- Une simulation en trois dimensions utilisant une matrice cubique de 1000 par 1000 par 1000 cellules nécessiterait un milliard de cellules de 16 octets, donc 16 Go.
- Une simulation moins ambitieuse en trois dimensions utilisant une matrice cubique de 300 x 300 x 300 cellules nécessiterait 27 millions de cellules de 16 octets, ce qui totaliserait environ 432 Mo.

Il faut également ajouter à cela les structures de données nécessaires pour l'algorithme d'identification des fragments (totalisant n^2 fois 32 bits) et quelques autres structures de données moins importantes (par exemple, les statistiques).

La simulation en trois dimensions est donc trop volumineuse pour les ordinateurs personnels sauf si l'on réduit la taille de la matrice. C'est pour cette raison qu'il serait avantageux à bien des égards de paralléliser le simulateur, notamment à cause des grandes quantités d'espace mémoire nécessaires mais aussi à cause des temps d'exécution beaucoup trop longs en trois dimensions. D'autre part, les automates cellulaires, utilisés pour la mise en oeuvre de plusieurs processus, sont par définition des systèmes naturellement parallèles (25; 2).

Il est important de noter qu'il aurait fallu fixer les exigences en mémoire propres au domaine de notre application mais que cela n'a pas été fait parce que nous nous étions basés sur un prototype qui, lui, n'en tenait pas compte.

3.1.2 Architecture

Le simulateur est divisé en sept modules qu'on peut voir au premier niveau du diagramme hiérarchique de la figure 3.1. Chacun des modules est décrit plus en détail à la section suivante. Pour mieux comprendre les relations entre les modules, on peut consulter le diagramme de classes en annexe B. Sur le diagramme de classes, chaque module est associé à une classe unique. On voit également des classes auxquelles ne sont associées aucun module. Ces classes supplémentaires sont nécessaires pour la mise en oeuvre mais pas pour la compréhension globale du système. C'est ce qui explique pourquoi on ne peut pas les voir sur le diagramme hiérarchique.

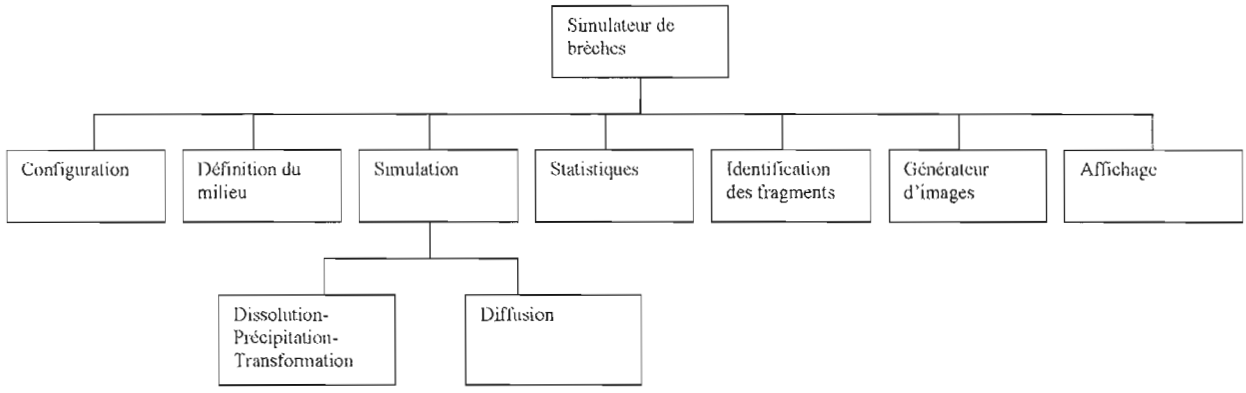


FIG. 3.1 Diagramme hiérarchique du logiciel de simulation de brèches hydrothermales.

Le logiciel est conceptuellement divisé en deux couches : la couche présentation et la couche application, aussi appelée le domaine. La couche présentation, dont les tâches sont essentiellement les interactions avec l'utilisateur et les entrées/sorties, comprend les modules d'affichage, de génération d'images, de statistiques et de configuration. La couche application comprend le module simulation, qui regroupe la mise en oeuvre de tous les processus. La figure 3.2 montre les modules du logiciel séparés selon leur couche respective.

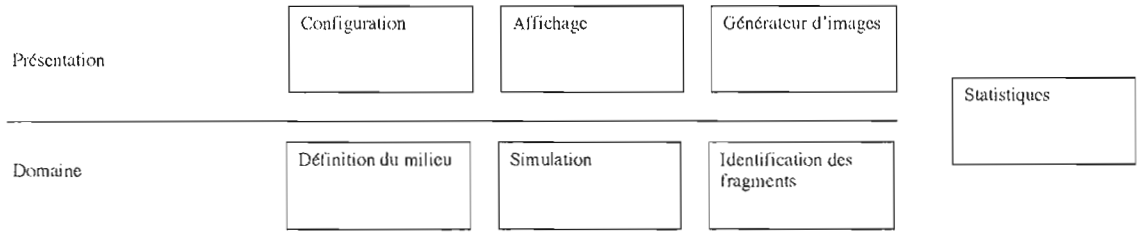


FIG. 3.2 Modules du simulateur séparés selon leur couche logiciel : en haut, les modules de la couche présentation et, en bas, les modules de la couche application, aussi appelée le domaine. Le module de statistique est à cheval sur les deux couches parce que celui-ci regroupe quelques variables (couche application) et quelques fonctions d'affichage(couche présentation).

3.2 Couche application

3.2.1 Définition du milieu

Le milieu, ou la roche virtuelle, est une matrice où chaque cellule peut prendre plusieurs états : minéralisé (quartz, feldspath, etc.) ou non minéralisé (connecté, non connecté). Les espaces non minéralisés sont remplis par un fluide déterminé au départ qui reste toujours le même. Un fluide aura un comportement différent selon qu'il est connecté ou non au réseau de fracture. Dans ce simulateur, deux cellules de fluide ne peuvent être connectées que par les plans orthogonaux et jamais par les diagonales. Les espaces minéralisés peuvent prendre autant d'états qu'il existe de minéraux dans la roche virtuelle. La figure 3.3 récapitule les types d'états possibles que peuvent prendre les cellules.

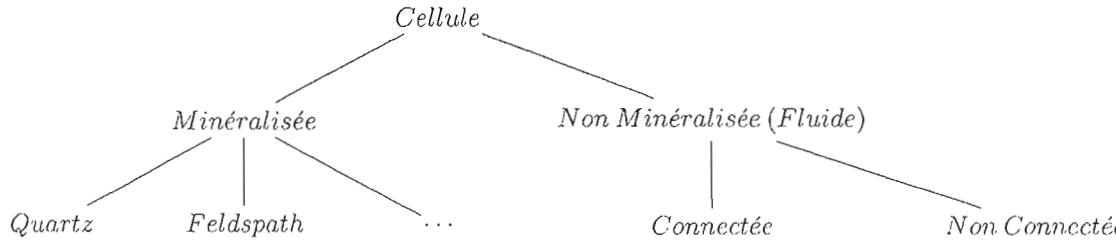


FIG. 3.3 Types de cellules possibles dans une roche virtuelle. Les types de cellules correspondent aux états d'un automate cellulaire.

Durant l'étape de configuration, l'utilisateur peut paramétrer la roche en précisant si la simulation se fera en deux ou trois dimensions, la taille de la matrice et la taille du fragment, le type de voisinage (von Neumann ou Moore) et, finalement, choisir entre trois roches prédéfinies ou importer la matrice d'une roche définie manuellement. Le rôle de l'étape de définition du milieu, une fois la configuration effectuée, est donc de générer la roche virtuelle qui sera traitée par le simulateur. La figure 3.4 montre un exemple de roche virtuelle.

3.2.2 Simulation

Le module de simulation modélise quatre des six processus géologiques associés à la formation des brèches. Trois de ces processus sont mis en oeuvre grâce des automates cellulaires : un de dissolution, un de précipitation et un d'échanges métasomatiques. L'autre processus géo-

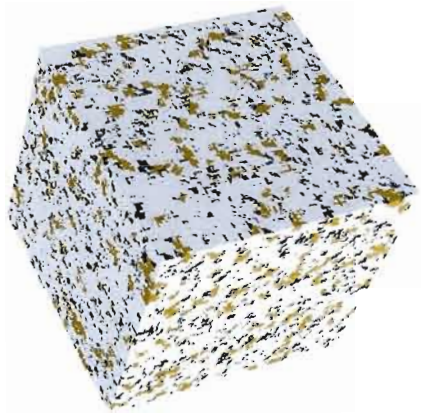


FIG. 3.4 Une roche plutonique virtuelle en 3D. On peut observer des phénocristaux de feldspath (gris), des micas (noir) et des quartz (blanc).

logique, la diffusion, est modélisé par une équation aux dérivées partielles et résolue par une méthode itérative. Les changements d'états des automates cellulaires, représentés à l'écran par des changements de couleur, correspondent soit à des changements de composition minéralogique (échanges métasomatiques), soit à un changement solide-fluide (dissolution) ou fluide-solide (précipitation). L'algorithme de simulation vu du plus haut niveau est le suivant :

```
PROCEDURE simuler( )
    POUR noIteration = 0, nbIterations FAIRE
        SI OPTION_DIFFUSION est activée ALORS
            diffuser()
        FIN SI
        changerEtats()
        // La méthode changerEtats() exécute le code des processus de dissolution,
        // de précipitation et de métasomatose.
    FIN POUR
FIN PROCEDURE simuler
```

Conceptuellement, les automates cellulaires utilisent une même matrice MA et l'algorithme de résolution de l'équation de la diffusion (ARED) possède la sienne, MD. Les cellules de la matrice des automates cellulaires possèdent les champs `nature` et `idFragment`. Celles de l'ARED possèdent seulement le champ `concentration`. Les matrices MA et MD sont de taille identique. Lorsque l'état d'une cellule i,j change sur la matrice MA, il faut parfois changer la concentration de la cellule correspondante sur la matrice MD. Cette situation survient lorsqu'il y a un changement d'état de solide à fluide ou de fluide à solide. En effet, lorsqu'une cellule

solide est dissoûte, sa concentration passe directement à 100 alors que lorsqu'il y a précipitation, sa concentration passe à 0.

Nous venons de voir que les changements d'état qui surviennent sur la matrice MA peuvent influencer les valeurs sur la matrice MD. Les valeurs de concentration de la matrice MD peuvent, quant à elles, influencer la manière dont surviendront les changements d'état sur la matrice MA. Le meilleur exemple est l'automate cellulaire d'échanges métasomatiques (voir section 3.2.2.4). Celui-ci peut seulement faire un échange de minéral si la concentration de la cellule de la matrice MD est supérieure au `seuilMetamorphose` définit pour une une cellule MA de cette nature. Autrement dit, le changement d'état pour une cellule $MA(i, j)$ dépend de la valeur de concentration de la cellule $MD(i, j)$.

Il serait plus simple de réunir les deux matrices en une seule. C'est effectivement ce que nous avons fait. Avec cette union, la matrice M possède donc les champs `nature`, `concentration` et `idFragment`. Le contexte est parfaitement propice à cette simplification car l'équation de la diffusion est très simple et est numériquement stable (voir chapitre 2) à ce niveau de précision. La majorité des autres systèmes de résolution d'équations numériques auraient nécessité d'être mis en oeuvre sur des matrices indépendantes avec des échelles qui leur soient propres (i.e., différentes de celles des automates cellulaires qui, dans ce cas-ci, utilisent toujours la même). C'est ce qu'il faudra faire avec la résolution de l'équation de fronts, qui sera vraisemblablement l'objet d'un autre projet de maîtrise. Le pseudo-code qui suit montre la boucle de parcours de la matrice (en deux dimensions) ainsi que les conditions déterminant le type de changement d'état, et ce pour une matrice M de taille N par N.

```
PROCEDURE changerEtats( )
    POUR i = 0 à N-1, j = 0 à N-1 FAIRE
            SI la nature de la cellule M(i, j) n'est pas un Fluide ALORS
                SI OPTION_DISSOLUTION est activée ALORS
                   dissoudre(i, j)
                FIN SI
                SI la cellule M(i,j) n'est pas dissoute ALORS
                     SI OPTION_TRANSFORMATION est activée ALORS
                         transformer(i, j)  // Métasomatose
                     FIN SI
                FIN SI
            SINON
                SI OPTION_PRECIPITATION est activée ALORS
                    precipiter(i, j)
                FIN SI
            FIN SI
    FIN POUR
FIN PROCEDURE changerEtats
```

Finalement, il est important d'expliquer comment le simulateur peut contrôler la vitesse de chaque processus. Tout d'abord, les paramètres géologiques de chaque processus permettent d'avoir un certain contrôle sur la vitesse. Cependant, il serait préférable d'avoir un paramètre vitesse indépendant car modifier les paramètres géologiques peut avoir une incidence sur le résultat final, c'est-à-dire la complexité morphologique, la forme, etc. En cela, la méthode de mise en oeuvre par *successive over-relaxation* de l'algorithme de la diffusion offre implicitement ce paramètre indépendant par la variable *omega*, qui contrôle la vitesse de convergence (voir aussi chapitre 2). On peut donc contrôler la vitesse de diffusion de l'eau dans la roche virtuelle. C'est le contrôle le plus rigoureux sur les vitesses qu'offre cette version du simulateur. Une meilleure solution consisterait à ajouter un paramètre de vitesse pour chaque processus et introduire une condition avant l'exécution de chaque processus. Par exemple, on pourrait utiliser l'expression `noIteration % parametreVitesseProcessusA` (% étant le symbole pour modulo). De cette façon, chaque processus pourrait être exécuté, de manière indépendante, une fraction des fois qu'on itère la boucle principale du simulateur.

3.2.2.1 Dissolution

Le processus géologique de dissolution est modélisé par un automate cellulaire. Cet AC est non déterministe, c'est-à-dire que pour chaque cellule et pour chaque changement d'état, un nombre pseudo-aléatoire[1] est généré et est utilisé pour déterminer l'état suivant. La règle locale de transition pour déterminer l'état suivant est décrite par la procédure suivante :

```
PROCEDURE dissoudre(int i, int j)
    déterminer la nature du minéral à dissoudre en M(i, j)
    indice_dissolution <- getTauxDissolutionMineral(i,j) * getTauxDissolutionAlpha(i,j)
    générer un nombre pseudo-aléatoire
    SI nombre aléatoire < indice_dissolution ALORS
        changer la nature de M(i, j) à Fluide
    FIN SI
FIN PROCEDURE dissoudre
```

Dans cette procédure, la fonction `getTauxDissolutionMineral()` retourne le taux de dissolution en fonction de la nature du minéral pour cette cellule. La valeur retournée par la fonction `getTauxDissolutionAlpha()` dépend du nombre de cellules de fluide voisines de cette

[1] Il faut normalement accorder une grande importance au choix du générateur de nombres aléatoires pour que le simulateur produise de bons résultats. Le choix de la méthode `random()` de la bibliothèque `Math` du langage Java semble produire de bons résultats.

cellule. Donc, selon le type de distribution choisi au moment de la configuration et du nombre de cellules de fluide voisines, la fonction retourne une valeur entre 0 et 1. Cette valeur, multipliée par le taux de dissolution minéralogique, donne l'indice de dissolution. L'indice de dissolution est en fait une valeur charnière ou un seuil relativement auquel est comparé le nombre pseudo-aléatoire. Lorsque le nombre aléatoire généré est inférieur ou égal à l'indice de dissolution, on dissoût ; lorsqu'il est supérieur, il n'y a aucun changement.

L'action de dissoudre revient en fait à changer le champ `nature`, ou si l'on préfère, l'état de la cellule, à la valeur `Fluide`.

3.2.2.2 Précipitation

Tout comme pour le processus de dissolution, le processus géologique de précipitation est modélisé par un automate cellulaire, lui aussi non déterministe.

La règle locale de transition pour déterminer l'état suivant est montré par la procédure suivante :

```
PROCEDURE précipiter(int i, int j)
    déterminer la nature du minéral à précipiter en M(i,j)
    indice_précipitation <- getTauxPrécipitationMinéral(i,j) * getTauxPrécipitationBeta(i,j)
    générer un nombre pseudo-aléatoire
    SI ce nombre aléatoire < indice_précipitation ALORS
        changer la nature de M(i, j) pour le minéral à précipiter
    FIN SI
FIN précipiter
```

Contrairement à la procédure de dissolution, on ne sait pas directement quel sera l'état suivant. Dans le cas de la dissolution, on sait toujours que si l'on dissoût, on change la nature de la cellule à l'état `Fluide`. Dans le cas de la précipitation, il faut déterminer quel type de minéral précipitera (fera son apparition). Plusieurs techniques auraient pu être utilisées. Nous avons choisi d'attribuer à chaque minéral un taux «d'apparition» qui pourrait être interprété comme la solubilité de ce minéral. Plus le taux d'apparition du minéral est élevé, moins le minéral est soluble. Les valeurs des taux d'apparition sont comprises entre 0 et 1. Une fonction calcule les nouveaux taux d'apparition en fonction du nombre de minéraux qui peuvent précipiter (le nombre de minéraux à précipiter dépend de la configuration du simulateur et, en particulier, du choix de la roche virtuelle). Ce calcul revient à établir une proportion entre les différents minéraux. Chaque minéral se voit attribué deux indices d'apparition (un intervalle) entre 0 et 1.

Un nombre pseudo-aléatoire est généré et on connaît le minéral à précipiter en identifiant entre quels indices se trouve le nombre pseudo-aléatoire généré. La figure 3.5 illustre de tels intervalles.

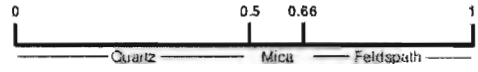


FIG. 3.5 Intervalles d'apparition attribués à chaque minéral.

Une fois les deux indices déterminés, la fonction `getTauxPrécipitationMineral()` retourne le taux de précipitation en fonction de la nature du minéral à précipiter, déterminé précédemment pour cette cellule. La valeur retournée par la fonction `getTauxPrécipitationBeta()` dépend quant à elle du nombre de cellules minérales voisines de cette cellule. Donc, selon le type de distribution choisi au moment de la configuration et du nombre de cellules minérales voisines, la fonction retourne une valeur comprise entre 0 et 1. Cette valeur, multipliée par le taux de précipitation minéralogique, donne l'indice de précipitation. Lorsque le nombre aléatoire généré est inférieur ou égal à l'indice de précipitation, on précipite ; lorsqu'il est supérieur, il n'y a aucun changement.

Comme pour la dissolution, l'action de précipiter revient en fait à changer le champ `nature` de la cellule, donc l'état de la cellule, à la valeur déterminée à l'aide du taux d'apparition.

3.2.2.3 Diffusion

Le processus de diffusion est modélisé par une équation aux dérivées partielles de type parabolique. Pour simplifier la mise en oeuvre de ce processus, nous avons supposé que le milieu est toujours homogène[2]. Cette simplification fait en sorte que nous pouvons utiliser une méthode itérative de résolution basée sur les différences finies. Dans le cas d'un milieu hétérogène, il aurait plutôt fallu utiliser la méthode par éléments finis (voir chapitre 2).

L'idée générale de l'algorithme de diffusion est simple : sur la matrice qui représente la roche virtuelle, on fixe les conditions aux limites, c'est-à-dire les cellules de bordure, et on définit des valeurs initiales pour les cellules internes. Puis, on applique la méthode itérative de résolution

[2]Dans ce simulateur, même quand le milieu n'est pas homogène, c'est-à-dire quand celui-ci est composé de plusieurs minéraux, les taux de diffusion sont identiques, quel que soit la nature du minéral.

de l'équation de la diffusion sur cette matrice. Les figures 3.6, 3.7, 3.8, 3.9, 3.10 montrent la matrice et, dans chaque cellule, les concentrations Cs (solide) et Cf (fluide) après les premières itérations. À noter que sur ces figures, on peut également observer les effets des autres processus qui agissent conceptuellement en simultané.

-	+	+	+	+	+	-
+	?	?	?	?	?	+
+	?	?	?	?	?	+
+	?	?	?	?	?	+
+	?	?	?	?	?	+
+	?	?	?	?	?	+
-	+	+	+	+	+	-

FIG. 3.6 Les cellules internes sont marquées par des '?' et changent de valeur dynamiquement. Les cellules de bordure sont marquées par des '+' et ont des valeurs fixes puisqu'elles représentent le fluide. Les cellules de coin sont marquées par des '-' et ne sont pas utilisées (25).

Il faut modéliser la diffusion des concentrations dans le fluide adsorbé[3] du solide. On considèrera que les éléments de bordure sont les cellules de fluide dont les concentrations restent stables (i.e., la concentration reste à 100% puisqu'elle représente un apport constant d'éléments). On suppose que les valeurs de bordures sont fixes parce que le système est ouvert. Dans un système ouvert, les éléments tels que le fluide sont libres d'entrer et sortir. Ceci a comme principale conséquence que le système n'atteindra jamais l'équilibre et qu'il ne manquera jamais de fluide ou de matière en solution. Les cellules internes sont les cellules minérales constituant le fragment de roche et leurs valeurs sont changeantes. Les bordures peuvent également se déplacer parce que l'état des cellules peut changer — il ne faut pas oublier que la méthode itérative est appliquée conceptuellement sur la même matrice qui sert à modéliser les autres processus géologiques, incluant des modèles basés sur des automates cellulaires. À chaque itération, on recalcule le taux de concentration dans le solide, où $C^t_{i,j}$, indique le taux de la cellule ij au temps t. Le nouveau taux de concentration est donné par la formule suivante pour une matrice bidimensionnelle :

$$C^{t+1}_{i,j} = (C^t_{i-1,j} + C^t_{i+1,j} + C^t_{i,j-1} + C^t_{i,j+1})/4 \tag{3.1}$$

[3] Adhésion physique ou physico-chimique à la surface d'un corps de substances en solution ou en suspension dans un fluide (43).

-	100	100	100	100	100	-
100	0	0	0	0	0	100
100	0	0	0	0	0	100
100	0	0	0	0	0	100
100	0	0	0	0	0	100
100	0	0	0	0	0	100
-	100	100	100	100	100	-

FIG. 3.7 Matrice représentant un fragment qui vient d'être immergé dans un fluide. La concentration du fluide (Cf) est constante et égale à 100%.

-	100	100	100	100	100	-
100	50	25	25	25	50	100
100	25	0	0	0	25	100
100	25	0	0	0	25	100
100	25	0	0	0	25	100
100	50	25	25	25	50	100
-	100	100	100	100	100	-

FIG. 3.8 Après la première itération, on peut observer que les concentrations internes sont modifiées au voisinage des bordures.

-	100	100	100	100	-	-
100	62.5	47.75	37.5	**56.25**	**100**	-
100	47.75	12.5	6.25	12.5	**56.25**	100
100	37.5	6.25	0	6.25	37.5	100
100	47.75	12.5	6.25	12.5	47.75	100
100	62.5	47.75	37.5	47.75	62.5	100
-	100	100	100	100	100	-

FIG. 3.9 À la deuxième itération, la diffusion se poursuit mais une cellule est dissoute (en haut à droite) entre les deux itérations, déplaçant la bordure vers l'intérieur.

-	100	100	100	100	100	-	-
100	100	62.5	47.75	37.5	56.25	100	-
100	100	47.75	12.5	6.25	12.5	56.25	100
100	100	37.5	6.25	0	6.25	37.5	100
100	100	47.75	12.5	6.25	12.5	47.75	100
100	100	62.5	47.75	37.5	47.75	62.5	100
100	100	**0**	100	100	100	100	-
-	100	100	100	100	100	100	-

FIG. 3.10 Toujours à la deuxième itération, mais suite à la dissolution, un minéral est précipité (en bas à gauche). Cet exemple montre aussi que le fragment est en réalité entouré de nombreuses couches de cellules de fluide, justement pour permettre au processus de précipitation d'avoir lieu. La véritable bordure ne change jamais d'état.

3.2.2.4 Échanges métasomatiques

Tout comme les processus de dissolution et de précipitation, le processus d'échanges métasomatiques est modélisé par un automate cellulaire non déterministe, c'est-à-dire que des nombres pseudo-aléatoires déterminent si, pour une cellule, il y aura échange métasomatique ou non. Ce processus est particulièrement intéressant parce qu'il est le seul qui, jusqu'à maintenant, utilise les résultats du processus de diffusion (voir section 3.2.2).

Il existe deux paramètres pour contrôler la vitesse des échanges métasomatiques. Le premier est la concentration de la cellule. Lorsque la valeur de la concentration dépasse le seuil de métamorphose intrinsèque du minéral de la cellule, alors celui-ci peut se transformer. L'autre paramètre est le taux de métamorphose. Celui-ci est obtenu en calculant le rapport entre la concentration et une constante de métamorphose. Cette constante de métamorphose n'a aucune représentation physique naturelle. Elle constitue uniquement un moyen de contrôler la vitesse du processus d'échanges métasomatiques. À noter que le mot métamorphose n'est pas très judicieusement choisi. Il s'agit vraiment d'un remplacement de minéral. L'algorithme d'échanges métasomatiques est le suivant :

```
PROCEDURE transformer(int i, int j)
    SI la concentration de M(i,j) > seuilMetamorphose du minéral de M(i,j) ALORS
        tauxMetamorphose <- concentration / CONSTANTE_METAMORPHOSE
        générer un nombre aléatoire
        SI  nombre aléatoire < tauxMetamorphose ALORS
            changer la nature de M(i, j) pour le minéral à transformer
            changer la concentration de M(i, j) au taux de départ du minéral à échanger
        FIN SI
    FIN SI
FIN tranformer
```

Des images du processus d'échanges métasomatiques sont disponibles au prochain chapitre.

3.2.3 Identification des nouveaux fragments

Comme nous l'avons vu au chapitre 1, lors de la dissolution, il peut arriver que se forment de nouveaux fragments à partir du fragment initial. Celui-ci peut se morceler en des parties de tailles semblables mais, plus fréquemment, laisser un grand nombre de petits fragments résiduels de petites tailles. Ce phénomène peut être observé visuellement mais il serait plus intéressant de pouvoir compiler des statistiques concernant ce phénomène. Pour ce faire, on peut utiliser ce qu'on appelle un algorithme d'identification des composantes connectées (*Connected Component Labeling*), algorithme initialement développé par Rosenfeld et Pfaltz pour le traitement d'images (44). Il existe plusieurs algorithmes dérivés de celui de Rosenfeld et Pfaltz mais ceux-ci sont tous plus complexes à mettre en oeuvre. L'algorithme de base sera donc utilisé. Lorsque l'option d'identification des fragments est sélectionnée, cet algorithme est exécuté une seule fois, à la fin de la simulation.

3.2.3.1 Voisinage et connectivité

Il faut au préalable définir quel sera le type de connectivité utilisé pour identifier les fragments. Les deux types les plus utilisés correspondent habituellement aux voisinages de von Neumann ou de Moore, respectivement qualifiés de connectivité-4 ($N_4(c)$) et connectivité-8 ($N_8(c)$). Plus formellement, on écrira qu'une cellule c possède quatre voisins directs orthogonaux $N_4(c)$ et quatre voisins diagonaux $N_D(c)$. Une cellule possédant huit voisins $N_8(c)$ consiste en l'union de $N_4(c)$ et de $N_D(c)$.

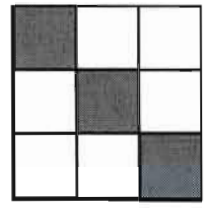


FIG. 3.11 Les cellules blanche forment un seul et même fragment par connectivité-8 mais deux par connectivité-4. Les cellules noires forment un seul fragment par connectivité-8 mais trois par connectivité-4. À noter qu'on n'utilise qu'un seul des deux systèmes à la fois, cette image servant seulement d'exemple pour montrer les deux types de connectivité.

Les deux types de voisinages sont imparfaits. Sur la figure 3.11, on peut voir qu'en utilisant la connectivité-8, toutes les cellules noires forment un fragment unique. Il en est de même pour les cellules blanches. En utilisant la connectivité-4, les cellules blanches du coin inférieur gauche forment un fragment différent des cellules blanches du coin supérieur droit. Les cellules noires sont toutes des fragments différents. On remarque que le choix du type de connectivité est critique pour déterminer le nombre de fragments du système. Nous avons choisi de mettre en oeuvre l'algorithme d'identification de fragments avec une connectivité-4 car cette approche semblait plus réaliste pour de la roche.

3.2.3.2 L'algorithme

L'identification des différents fragments se fait en deux passes.

Soit une matrice M de taille N par N :

1- Première passe : on parcourt toutes les cellules de la matrice M, rangée par rangée ($0..N$ - 1) et colonne par colonne ($0..N$ - 1)

- On vérifie tous les voisins de la cellule $M(i,j)$.
- Si la cellule $M(i,j)$ n'a aucun voisin connecté, on crée un nouvel identifiant unique et on l'attribue à la cellule.
- Si la cellule $M(i,j)$ a exactement un voisin connecté avec le même identifiant qu'elle, on lui attribue cet identifiant.
- Si la cellule $M(i,j)$ possède deux cellules ou plus connectées mais pas nécessairement avec le même identifiant, on choisit et on attribue à la cellule $M(i,j)$ l'un de ces identifiants et on prend note que tous ces identifiants sont maintenant équivalents.

2- Seconde passe : on parcourt toute la matrice comme à la première passe pour résoudre les équivalences.

Les équivalences sont habituellement conservées dans deux tables. La première table contient toutes les équivalences d'identifiants dans une structure de type dictionnaire appelée `TreeMap`. Il s'agit en fait d'une structure hybride entre une table de hachage et un arbre de recherche. Pour être plus précis, chaque cellule du tableau contient une nouvelle instance de type `TreeMap`. La seconde table contient l'identifiant minimal (`int`) pour tous les identifiants de chaque dictionnaire du premier tableau. Pour une matrice M de taille N^2, on peut définir que la taille de chaque tableau sera également N^2.

La complexité asymptotique de cet algorithme est au maximum $O(N^3)$. Le pire cas imaginable est celui où les dictionnaires de chaque cellule de la table d'équivalence seraient de taille N.

3.3 Couche présentation

Certains modules, par exemple celui de configuration, permettent à l'utilisateur d'entrer des données. D'autres, comme celui des statistiques, le générateur d'image ou l'affichage servent plutôt à présenter des résultats. Il s'agit de modules de sortie de données.

3.3.1 Entrées de données ou «module de configuration»

Le module de configuration fait deux choses : il lit un fichier définissant les minéraux qui pourront être utilisés dans le simulateur et il lit un fichier de configuration généré par l'interface graphique. Ce fichier de configuration est aussi modifiable à la main. L'interface graphique est un formulaire écrit en HTML et en Javascript. L'utilisateur peut également confi-

gurer les minéraux. Les champs configurables des minéraux sont : `couleur`, `idMetamorphose`, `seuilMetamorphose`, `tauxMetamorphose`, `tauxDissolution`, `tauxPrécipitation` et `ratioApparition`.

Les informations envoyées par le formulaire sont traitées côté serveur par du code PHP. Le nom de chaque champ du formulaire HTML est inscrit dans le fichier de configuration avec sa valeur. Le simulateur lit le fichier de configuration et place chaque champ dans une structure de données de type dictionnaire. Nous avons utilisé une structure appelée `TreeMap` (`java.util.TreeMap`). Puis, une méthode affecte les valeurs de chaque clé du dictionnaire aux variables de la classe `Parametres`, laquelle a pour rôle de rassembler en un seul endroit tous les champs configurables et les constantes pour l'ensemble du simulateur.

3.3.2 Statistiques

Le module de statistiques est une classe Java contenant plusieurs variables définies à la section 4.1.2. Une méthode permet l'affichage à l'écran et une autre permet d'écrire les données dans un fichier.

3.3.3 Affichage

Ce module s'occupe d'afficher à l'écran la matrice ou des coupes de la matrice (lorsque celle-ci est en trois dimensions). Les bibliothèques utilisées sont `javax.Swing` et `java.awt`. Trois fenêtres apparaissent lorsque la simulation se termine : une fenêtre montrant les taux de concentrations du processus de diffusion avec un dégradé de couleurs, une fenêtre montrant la natures des cellules (états) et une fenêtre montrant les différents fragments avec une couleur différente pour chaque identifiant de fragment. À noter que l'option de création d'images en noir et blanc s'applique seulement sur les images de format GIF archivées à chaque itération, et non sur les trois fenêtres Java. Ce sera le sujet de la prochaine sous-section.

3.3.4 Générateur d'images

Le générateur d'image est une classe appelée `ImageGenerator` possédant deux groupes de méthodes. Le premier groupe a pour fonction de créer des images couleur et le second de créer des images en noir et blanc. Chaque groupe est constitué des deux mêmes méthodes surchargées : l'une prend un objet de type I_2D en paramètre, et l'autre un objet de type

I_3D (où I_2D et I_3D sont des interfaces Java utilisées selon le nombre de dimensions choisi). Cette classe utilise la bibliothèque `Acme.JPM.Encoders.GifEncoder` pour encoder les images en format GIF. Les méthodes qui créent les images en couleur définissent les couleurs en fonction du champ `nature` de chaque cellule. Les couleurs attribuées à chaque valeur possible du champ `nature` sont enregistrées dans la classe `Parametre`. Lorsque la méthode doit traiter une matrice en trois dimensions, celle-ci encode une image correspondant à une coupe orthogonale du cube.

3.4 Langage et technologies

Le simulateur est codé en Java et utilise la bibliothèque externe `Acme.JPM.Encoders.GifEncoder` pour créer les images de la matrice. L'interface graphique pour la configuration est en format Html et en javascript. Le fichier de configuration est généré par du code PHP. On peut voir, en annexe A, les fenêtres de configuration du simulateur de brèches.

Chapitre IV

UTILISATION DU SIMULATEUR DE BRÈCHES HYDROTHERMALES

Le présent chapitre explique comment un utilisateur peut interagir avec le simulateur. Nous verrons qu'il est possible d'entrer des données par un formulaire qui génèrera un fichier de configuration de format approprié. Nous verrons aussi que les données de sortie peuvent être de types variés (i.e., fenêtres à l'écran, fichiers d'image, fichiers de texte). Ce chapitre est donc une sorte de manuel d'utilisation du simulateur. Le formulaire est disponible à l'adresse suivante : `http ://www.martinlalonde.ca/simulateur/simulateur.html`. La figure 4.1 illustre une utilisation typique du simulateur du début jusqu'à la fin.

4.1 Description

4.1.1 Configuration

La première étape d'une simulation est toujours la configuration. Un formulaire en HTML dynamique permet de définir les différents paramètres. Une simulation typique est présentée en guise d'exemple à la section 4.2. Si l'utilisateur suit l'ordre logique de présentation du formulaire, la première étape consiste à configurer les paramètres de structure de l'automate cellulaire. Ces paramètres sont le nombre de dimensions, la taille de la matrice et du fragment, le type de voisinage (graphiquement explicite) et le nombre d'itérations. Ensuite, l'utilisateur définit la roche virtuelle ainsi que plusieurs paramètres de simulation décrits plus loin par des tableaux et commentés par un exemple d'utilisation à la section 4.2. La dernière étape consiste à définir les propriétés des minéraux. Un graphique présentant le formulaire de configuration est présenté en annexe A. Les paramètres à configurer propres à la simulation, les options et les propriétés des minéraux sont décrits respectivement dans les tableaux 4.1, 4.2, 4.3.

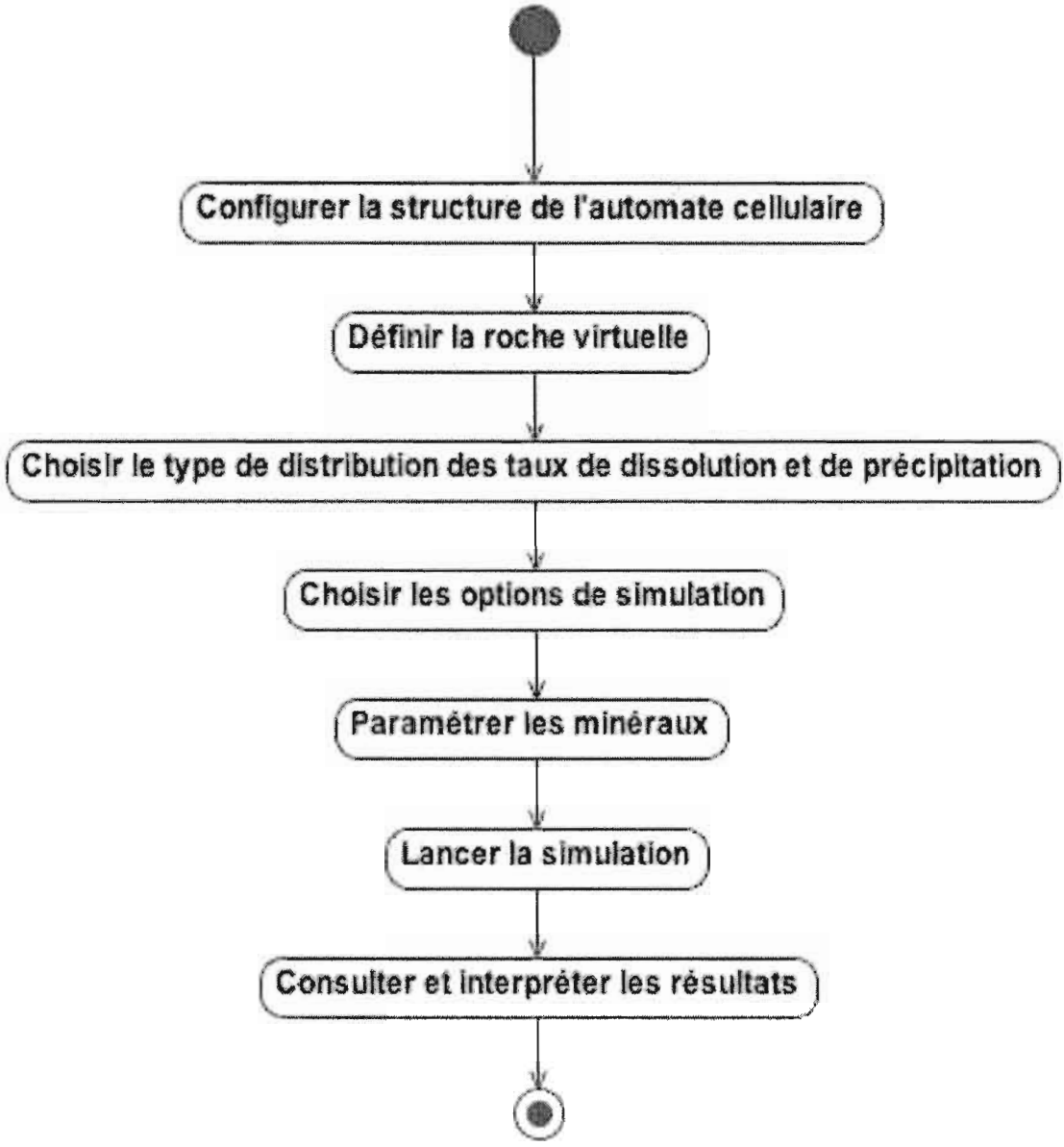


FIG. 4.1 Diagramme d'activités décrivant l'utilisation du simulateur, incluant l'étape de configuration, de simulation et de consultation des résultats.

Paramètre	Description
Nombre de dimensions	La matrice sera-t-elle à deux ou trois dimensions ?
Taille de la matrice	Taille de la matrice, que celle-ci soit en 2D ou en 3D.
Taille du fragment	Taille du fragment de roche à l'intérieur de la matrice. La taille du fragment doit toujours être inférieure à la taille de la matrice et toutes les autres cellules prennent l'état de fluide.
Type de voisinage	Voir la section 2.2.4.
Nombre d'itérations	Le nombre de fois qu'on désire exécuter chacune des étapes de simulation.
Roche virtuelle	L'utilisateur doit choisir entre trois roches prédéfinies par le simulateur ou importer un fichier de roche virtuelle. Ce fichier peut soit être le résultat d'une simulation antérieure, soit une roche virtuelle générée par un outil extérieur. Les trois roches virtuelles prédéfinies sont une roche monominéral, une roche biminéral et une roche triminéral.
Porosité	L'utilisateur peut choisir le pourcentage de porosité de la roche virtuelle qu'il désire traiter. Ce paramètre est seulement valable si l'utilisateur choisit une des trois variétés de roche virtuelle. Si l'utilisateur utilise un fichier de données contenant la matrice, le paramètre de porosité n'est pas utilisé. La porosité peut prendre des valeurs réelles entre 0 et 100. Une valeur réaliste se situe plutôt en dessous de 1.
Distribution des taux de dissolution	Le choix du type de distribution des taux de dissolution influencera la forme que prendra le fragment de roche à la fin de la simulation. Le type de distribution définit la règle locale de transition de l'automate cellulaire de dissolution. L'utilisateur peut choisir quatre types de distribution : constant, log-normal (voir chapitre 1), linéaire ou arbitraire (choix manuel de la distribution).
Distribution des taux de précipitation	Le choix du type de distribution des taux de précipitation influencera la forme que prendra le fragment de roche à la fin de la simulation. Le type de distribution définit la règle locale de transition de l'automate cellulaire de précipitation. L'utilisateur peut choisir quatre types de distribution : constant, log-normal (voir chapitre 1), linéaire ou arbitraire (choix manuel de la distribution).

TAB. 4.1 Les paramètres de configuration de la simulation

Paramètre	Description
Diffusion	L'utilisateur peut configurer le paramètre *omega* (voir chapitre 2) qui contrôle la vitesse de convergence de l'algorithme de diffusion. L'utilisateur peut demander de calculer le *omega* optimal. La valeur d'*omega* devrait se situer entre 0 et 2.

4.1.2 Statistiques

Les informations conservées pour chaque simulation sont décrites dans le tableau 4.4.

4.1.3 Affichage à l'écran et historique des images

L'affichage à l'écran se résume à trois fenêtres. La première fenêtre montre le fragment avec des couleurs distinctives selon la nature (état) des cellules. La seconde fenêtre montre la diffusion, c'est-à-dire les concentrations selon un dégradé de couleur. La troisième fenêtre montre chaque fragment identifié grâce à une couleur différente. Ces trois fenêtres montrent le résultat final de la simulation. Les figures 4.2, 4.3 et 4.4 montrent les trois fenêtres après une simulation typique.

Pour voir les phases intermédiaires de la simulation, le simulateur offre un mécanisme de création d'un historique d'images en couleur ou noir et blanc. L'historique en couleur enregistre seulement la fenêtre de la nature des cellules. L'historique des images en noir et blanc distingue uniquement les cellules minérales en noir et le fluide en blanc. Ces images, de format GIF, sont utilisées par un logiciel externe appelé NIH pour calculer la dimension fractale des fragments.

Paramètre	Description
Option de dissolution	Lorsque cette option est sélectionnée, le simulateur active l'automate cellulaire de dissolution.
Option de précipitation	Lorsque cette option est sélectionnée, le simulateur active l'automate cellulaire de précipitation.
Option d'échanges métasomatiques	Lorsque cette option est sélectionnée, le simulateur active l'automate cellulaire d'échanges métasomatiques.
Option de diffusion	Lorsque cette option est sélectionnée, le simulateur résout l'équation de la diffusion.
Option d'identification des fragments	Lorsque cette option est sélectionnée, le simulateur exécute la procédure d'identification des fragments. Cette procédure est exécutée une seule fois à la fin de la simulation.
Option de déboggage	Pendant l'exécution, lorsque cette option est sélectionnée, le simulateur imprime à l'écran diverses informations concernant l'exécution.
Option d'impression de statistiques	Lorsque cette option est sélectionnée, le simulateur imprime dans un fichier les statistiques récoltées pendant la simulation. Si ce mode est désactivé, le simulateur imprime les statistiques à l'écran seulement.
Option de création d'un fichier de sortie	Lorsque cette option est sélectionnée, à la fin de la simulation, un fichier représentant la matrice est créé. Ce fichier peut être utilisé pour simuler d'autres phases d'altérations sur un même fragment.
Option de création d'un historique d'images en noir et blanc	À chaque itération, une image GIF en noir et blanc de la matrice est créée (une tranche pour la matrice en trois dimensions). Le noir représente la roche, tous minéraux confondus, et le blanc représente le fluide. À noter qu'il serait plus intuitif d'attribuer la couleur noir au fluide et la couleur blanc aux minéraux. Toutefois, nous produisons des images en noir et blanc pour calculer leur complexité morphologique à l'aide du logiciel NIH. Celui-ci demande que les couleurs soient attribuées de cette façon.
Option de création d'un historique d'images en couleur	À chaque itération, une image GIF en couleur de la matrice est créée (une tranche pour la matrice en trois dimensions). Chaque couleur est associée à un minéral. Par défaut, le noir représente l'absence de minéraux, donc la présence de fluide.

TAB. 4.2 Les options de simulation

Paramètre	Description
idMetamorphose	Numéro du minéral qui remplace le minéral courant lors des échanges métasomatiques.
seuilMetamorphose	Seuil de concentration à partir duquel le mécanisme d'échanges métasomatiques peut commencer. Les concentrations sont en pourcentages (0-100) donc les valeurs du seuil doivent être comprises entre 0 et 100.
tauxMetamorphose	Taux intrinsèque au minéral qui régit la vitesse de remplacement pour les échanges métasomatiques. La valeur doit toujours être entre 0 et 1.
tauxDissolution	Taux intrinsèque au minéral qui régit la vitesse de dissolution. La valeur doit toujours être entre 0 et 1.
tauxPrecipitation	Taux intrinsèque au minéral qui régit la vitesse de précipitation. La valeur doit toujours être entre 0 et 1.
tauxApparition	Valeur entre 0 et 1 indiquant la probabilité d'apparition d'un minéral par rapport aux autres minéraux présents lorsqu'il y a une possibilité de précipitation. Par exemple, si les minéraux présents sont le quartz, le feldspath et le minéral transitoire 1 (voir l'annexe A) et que tous ont un ratio d'apparition de 0.5, tous les minéraux ont une chance égale de précipiter avant même de considérer le taux de précipitation de chacun. À noter que les ratios d'apparition sont indépendants les uns des autres. Pour de plus amples informations concernant l'algorithme de précipitation, référez-vous au chapitre 3.

TAB. 4.3 Propriété des minéraux

Information	Description
Nombre de fragments	Le nombre de fragments identifiés à la fin de la simulation.
Nombre de cellules selon la nature	Le nombre de cellules pour chaque état du champ `nature` à la fin de la simulation.
Nombre de cellules dissoutes	Le nombre total de cellules dissoutes pendant toute la simulation.
Nombre de cellules précipitées	Le nombre total de cellules précipitées pendant toute la simulation.
Nombre de cellules remplacées par échange métasomatique	Le nombre total de cellules remplacées par échange métasomatique pendante toute la simulation.

TAB. 4.4 Statistiques reccueillies durant la simulation

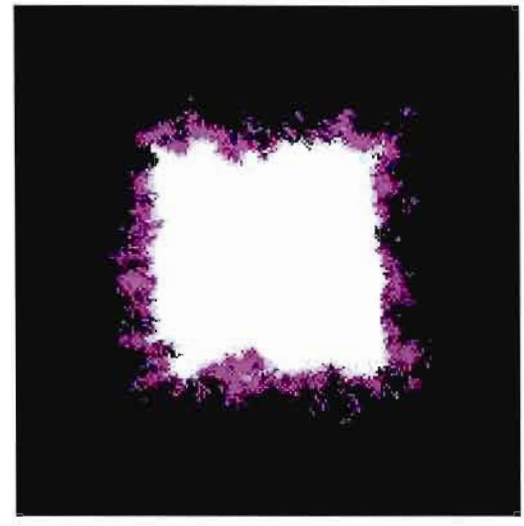


FIG. 4.2 Fenêtre d'affichage de la nature des minéraux. On distingue, en blanc, le minéral primaire et, en magenta, le minéral précipité.

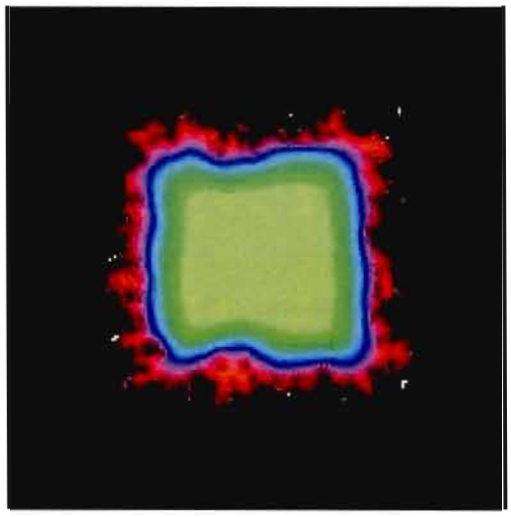


FIG. 4.3 Fenêtre d'affichage des taux de concentration. Les taux de concentration sont décroissants de l'extérieur vers l'intérieur du fragment.

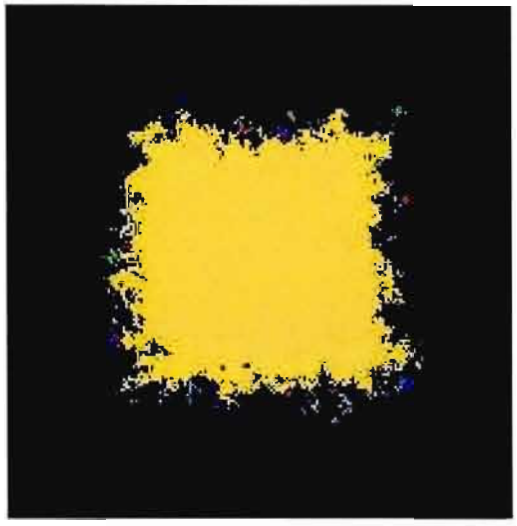


FIG. 4.4 Fenêtre d'affichage des fragments identifiés. Chaque fragment est représenté par une couleur différente.

4.2 Exemple d'utilisation

Résumons le fonctionnement du simulateur par un exemple d'utilisation. On suppose que l'utilisateur est un géologue. Rappelons-nous qu'une utilisation possible du simulateur est d'aider à comprendre quels sont les mécanismes qui ont pu engendrer la roche qu'on observe dans la nature. C'est ce que notre géologue virtuel tentera de faire pour les fins de l'exemple.

Configurer la structure de l'automate cellulaire et définir la roche virtuelle

Supposons que celui-ci veuille tenter d'obtenir une roche à deux minéraux initiaux, aux formes arrondies, à la surface lisse et composée d'un troisième minéral, essentiellement observable à sa surface. Il choisit d'abord les paramètres de structure : le nombre de dimensions sera de deux parce qu'on juge qu'une vue en 3D n'est pas nécessaire et on ne veut pas que la simulation dure trop longtemps. On suppose qu'une taille de 300 pour la matrice et de 250 pour le fragment fera l'affaire. Le type de voisinage sera Moore car, comme nous le verrons au chapitre 5, nous savons que ce type de voisinage donne de meilleurs résultats que le voisinage de von Neumann pour l'arrondissement des fragments. Le nombre d'itérations sera fixé arbitrairement à 100 (on ne sait pas encore si la simulation demandera plus ou moins d'itérations). Ensuite, il lui faut soit définir sa propre roche virtuelle à partir d'un canevas prédéterminé, soit choisir un fichier qui contient une matrice déjà existante. Le géologue choisit donc un bloc à deux minéraux initiaux avec une porosité de 1%.

Choisir le type de distribution des taux de dissolution et de précipitation

Il faut maintenant choisir la distribution des taux de dissolution et de précipitation en fonction du voisinage. La distribution de type log-normal donne de bons résultats pour l'arrondissement des fragments. Le formulaire permet d'ajuster manuellement les valeurs de la courbe de distribution. Des valeurs sont suggérées mais c'est seulement à titre d'exemple et il ne faut pas se limiter à celles-ci. Pour sa première expérience, le géologue choisira de ne pas inclure le processus de précipitation dans sa simulation. Il est vrai que le comportement du système devient alors beaucoup plus complexe et imprévisible. Si le géologue désire inclure des phénomènes de métasomatose dans sa simulation, il devra également choisir d'inclure le processus de diffusion et définir une vitesse pour le processus. Le géologue juge que le troisième minéral — le minéral de surface — devrait être obtenu par métasomatose. Il choisit donc un *omega* optimal, calculé automatiquement et permettant d'obtenir la vitesse de convergence la plus rapide.

Choisir les options de simulation

À ce stade, il reste trois étapes de configuration avant la simulation : les options de simulation, c'est-à-dire, les processus et les mécanismes qui seront activés, les autres options qu'on pourrait qualifier de «support» et enfin, la configuration des minéraux et de leurs propriétés. En ce qui concerne les options de simulation, le géologue choisit de simuler la dissolution, les échanges métasomatiques ainsi que la diffusion [1]. Notons que le géologue a choisi de ne pas activer le mécanisme d'identification de nouveaux fragments car cette information n'apporterait rien d'utile ou du moins, pas si l'on tente d'obtenir des fragments lisses et arrondis. Il doit ensuite configurer l'avant dernière catégorie d'options. Il décide d'imprimer les statistiques dans un fichier, de créer une matrice de sortie, de créer un historique d'images en noir et blanc et un historique d'images en couleur.

Paramétrer les minéraux

Il ne reste plus qu'à définir les propriétés des minéraux — les propriétés de la nature des cellules serait plus juste — utilisés dans la roche virtuelle. Le géologue avait préalablement choisi un bloc à deux minéraux initiaux. Il ne devrait pas être nécessaire de modifier la porosité, le fluide et la bordure. Les deux minéraux initiaux sont les deux premiers sur la liste. S'il y a précipitation ou métasomatose, il faut considérer les minéraux transitoires. Il choisit d'attribuer un taux de dissolution identique aux deux minéraux initiaux et un taux plus faible pour le minéral transitoire, résultat de la métasomatose. Il choisit également des taux de métamorphoses identiques pour les minéraux initiaux et un taux de métamorphose nul pour le minéral transitoire (le minéral transitoire ne donne pas lieu à des échanges métasomatiques).

Lancer la simulation, consulter et interpréter les résultats

La phase de configuration étant terminée, le géologue envoie sa requête et un fichier de configuration est généré. Il peut alors le télécharger sur son ordinateur et le placer dans le dossier racine du simulateur. Si nécessaire, il peut aisément faire des changements au fichier de configuration car celui-ci est de format texte et les noms des champs sont évocateurs. Le géologue lance alors la simulation. Après un certain temps, qui dépend des paramètres de configuration, mais plus spécialement, de la taille de la matrice, du nombre de dimensions et du nombre d'itérations, les trois fenêtres d'affichage apparaissent à l'écran. Pour voir les phases intermédiaires,

---

[1] À noter que si le géologue avait choisi de simuler le processus d'échanges métasomatiques sans le processus de diffusion, il n'y aurait pas eu de métasomatose.

il faut consulter l'historique des images. Seule la fenêtre de la couleur en fonction de la nature est conservée dans l'historique des images en couleur. Le géologue peut également consulter le fichier des statistiques qui présentent de manière numérique le résultat de la simulation. Les statistiques conservées sont décrites à la section 4.1.2. Finalement, il est possible de recommencer une toute nouvelle simulation, avec des paramètres complètement différents, commençant là où s'est terminée une simulation précédente. Il suffit de choisir l'option d'importer un fichier de roche virtuelle au moment de définir la roche.

Chapitre V

EXPÉRIMENTATION ET RÉSULTATS

Ce chapitre est consacré d'abord à la présentation des résultats obtenus à l'aide d'un prototype du simulateur actuel, résultats qui donnèrent lieu à une publication (32). Ce sont ces expériences, ainsi que leurs résultats, que nous présentons dans la première partie du chapitre. Quelques autres résultats, tels que le choix du voisinage et le choix du treillis, sont des conclusions obtenues grâce aux expériences sur le nouveau simulateur. Dans la seconde partie du chapitre, nous présentons quelques exemples de simulations possibles ainsi qu'une brève description de quelques expériences qu'il faudrait effectuer pour valider plus à fond le simulateur. Quelques méthodes sont proposées.

5.1 Prototype du simulateur et expérimentation

Il faut savoir que, préalablement à la conception du simulateur de brèches hydrothermales, existait un prototype, un simulateur de dissolution, qui servit de base à la conception du simulateur actuel. Ce prototype avait été conçu par l'auteur dans le cadre d'un stage de recherche, préalablement à la maîtrise. Grâce à ce prototype, il fut possible d'effectuer quelques expériences, dont les résultats et interprétations sont en voie de publication. Nous savons maintenant que notre modèle de dissolution, basé sur un automate cellulaire, est valide. Ces expériences furent exécutées et validées par des géologues. D'ailleurs, il est pertinent de mentionner que toutes les tâches de validation effectuées ou à effectuer, ont été ou le seront par des géologues.

5.2 Résultats validés

5.2.1 Choix du régime de dissolution

Au chapitre 1, nous avons vu que les géologues distinguent deux régimes de dissolution : le régime cinétique et le régime DLR. On qualifie le premier de cinétique parce que la vitesse de dissolution est limitée uniquement par la vitesse de réaction chimique qui corrode la surface d'un fragment. Ce régime est donc indépendant de la surface exposée. Le résultat est un fragment dans un état hautement accidenté. L'autre régime est qualifié de «limité par la diffusion» (*Diffusion Limited Regime* — DLR) parce que seules les parties les plus exposées du solide sont atteintes par le fluide. Le taux de dissolution du régime DLR est donc dépendant de la surface exposée au fluide et est caractérisé par l'arrondissement et le lissage du solide.

La forme des fragments a été étudiée dans de nombreux domaines des sciences de la terre, incluant la sédimentologie, la géologie de surface, la géologie économique, la volcanologie et la pétrologie (32). L'arrondissement des fragments a été fréquemment observé et est généralement attribué à l'abrasion pendant le transport. Notre simulateur a permis de démontrer que la complexité de la surface des fragments peut également être reliée au régime de dissolution. Nous avons vu au chapitre 1 que la dimension fractale pouvait être utilisée pour mesurer la complexité morphologique des fragments de roche dans les brèches. Un étudiant de géologie a été engagé suite au développement du prototype et a testé le simulateur en faisant varier différents paramètres : la composition, la porosité, la distribution des probabilités de dissolution en fonction de l'exposition au fluide. Ces expériences nous ont permis d'observer un arrondissement lorsque la fonction de la probabilité de dissolution en fonction de la surface exposée ressemble à une fonction log-normale. En fait, il suffit d'attribuer une probabilité de dissolution nulle lorsque le voisinage d'une cellule est de configuration stable (parce que le cercle et la sphère sont les formes les plus stables dans la nature[1]). Les autres types de fonctions donnent en général une variante quelconque (et *a priori* peu intéressante) du régime cinétique. Une configuration stable correspond à une seule cellule voisine de fluide avec un voisinage de von Neumann et trois cellules voisines de fluide (ou moins) avec un voisinage de Moore (voir également la figure 5.2). Pour mieux comprendre cette notion de stabilité, on peut imaginer qu'un cercle est en fait composé d'une infinité de droites infinitésimales, tangentes au centre du cercle. Si l'on regarde le périmètre d'un cercle de

[1]On ne peut pas aplatir plus un cercle ou une sphère sans les déformer, sans les rendre moins plats ailleurs. On peut penser aussi à la façon dont les corps célestes (planètes, étoiles, etc.) se stabilisent.

vraiment très proche, on a l'impression que le périmètre du cercle est parfaitement droit (un peu comme on a l'impression que la terre est plate parce qu'elle est en réalité une gigantesque sphère).

Finalement, les expériences ont permis de montrer que d'autres paramètres tels que la porosité et la composition de la roche ont une influence mineure sur la forme finale des fragments en dissolution. Ces facteurs influencent principalement la vitesse de dissolution, et non la forme que ceux-ci auront à la fin.

5.2.2 Choix du voisinage

Au chapitre 2, nous avons vu que le voisinage local d'une cellule pouvait varier d'un automate cellulaire à l'autre. Le choix du type de voisinage a plusieurs conséquences sur la simulation. Rappelons que le voisinage de von Neumann inclut seulement les cellules voisines qui sont dans les plans ortogonaux alors que le voisinage de Moore inclut en plus les diagonales.

La première conséquence du choix d'un type de voisinage est l'arrondissement des fragments. En effet, après plusieurs simulations en 2D, nous en sommes arrivés à la conclusion que le voisinage de von Neumann peut difficilement modéliser correctement le processus de dissolution. Observons d'abord les deux figures suivantes :

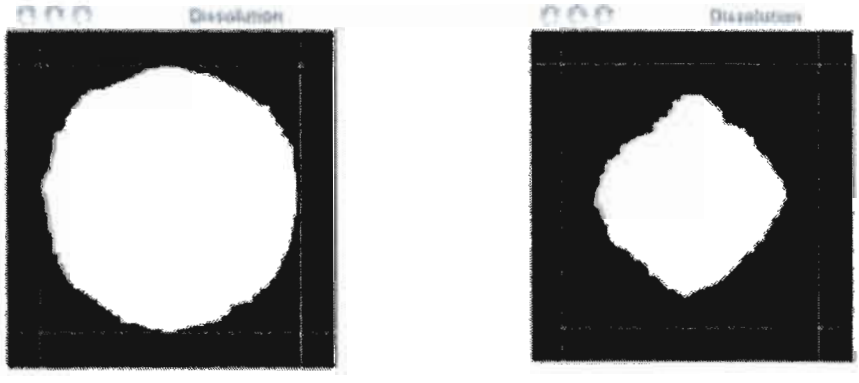


FIG. 5.1 À gauche : après 200 itérations. À droite : après 300 itérations. Distribution des probabilités log-normale.

Après un certain nombre d'itérations, les coins seront arrondis. Toutefois, si l'on continue à dissoudre, les coins seront plutôt aplatis. C'est ce qu'on observe sur la figure 5.1. La cause est simple : on indique au simulateur que la probabilité de dissolution est nulle lorsqu'une cellule possède une seule cellule voisine dont la nature soit un fluide. Cette conception est incorrecte avec un voisinage de von Neumann, qui ne tient pas compte des diagonales. En effet, si l'on considère que les plans orthogonaux sur un cercle correspondent à $\pi/2$, π, $3\pi/2$ et 2π et que les diagonales correspondent $\pi/4$, $3\pi/4$, $5\pi/4$ et $7\pi/4$, alors les deux groupes de cellules de la figure 5.2 représentent de gauche à droite, les positions $3\pi/2$ et $5\pi/4$. Or, on voit bien que pour un voisinage de von Neumann, le nombre de cellules voisines change dépendamment d'où on se trouve sur le cercle : un seul voisin à $3\pi/2$ et deux voisins à $5\pi/4$. Normalement la surface exposée au fluide devrait être à peu près partout la même lorsque le fragment est de forme circulaire ou sphérique (en 3D). Or ce n'est clairement pas le cas avec un voisinage de von Neumann.

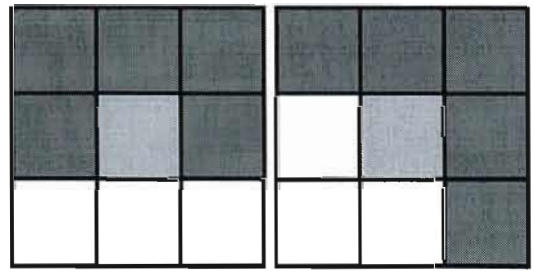


FIG. 5.2 Sur les deux figures, on peut voir la représentation d'une droite sur une matrice composée de cellules carrés à deux endroits sur le périmètre d'un cercle : à gauche à $3\pi/2$ et à droite à $5\pi/4$. Les cellules en gris foncé représente des cellules minérales, la cellule en gris pâle est la cellule à dissoudre et les cellules blanches représentent des cellules de fluide.

Avec un voisinage de Moore, le nombre de cellules voisines dont la nature est un fluide à $3\pi/2$ et $5\pi/4$ est le même, c'est-à-dire trois. Étant donné que le cercle est la morphologie la plus stable et équilibrée que l'on puisse obtenir, il en découle qu'on doit attribuer la probabilité 0 à notre fonction *Alpha* lorsque qu'il y a un, deux ou trois cellules de fluide voisines à une cellule minérale. Maintenant observons la figure 5.3. Ces fragments semblent mieux arrondis que ceux produits avec un voisinage de von Neumann. Toutefois, ce n'est pas parfait. On peut clairement observer la formation progressive d'un fragment de forme octogonale (8 côtés). On fait facilement le lien avec le voisinage de Moore car le nombre de voisins pour ce type de voisinage est huit. Avec le voisinage de von Neumann, dont la caractéristique est de tenir compte uniquement

des voisins dans les plans orthogonaux, le fragment se transformait progressivement en losange (quatre côtés).

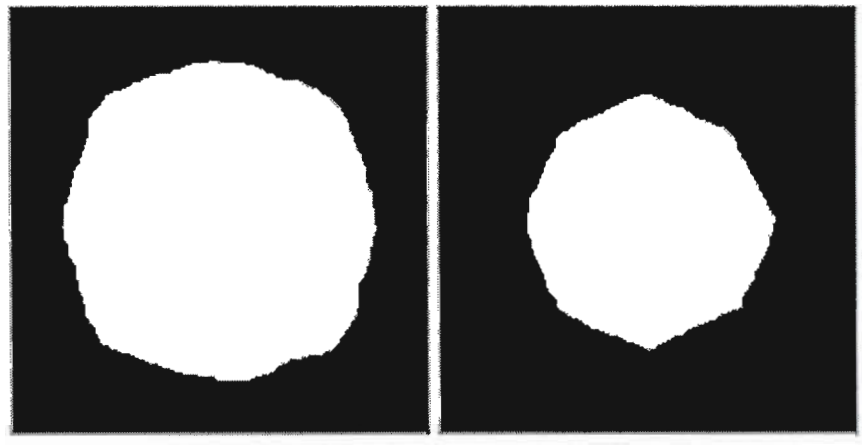


FIG. 5.3 À gauche : après 160 itérations. À droite : après 300 itérations. Distribution des probabilités log-normale.

Les imperfections que nous avons pu observer pour les deux types de voisinages nous amènent à penser que nous avons probablement atteint une limite. Cette limite correspond à la forme implicite d'une cellule, c'est-à-dire le carré ou le cube (en 3D). La forme d'une cellule dépend du treillis choisi pour la matrice. Le carré est la forme la plus facile à mettre en oeuvre parce que l'ordinateur permet d'enregistrer l'information sous forme de tableaux qui sont organisés en matrices dont les cellules sont des carrés. Il existe toutefois des treillis de type triangulaires et hexagonaux. Les treillis hexagonaux sont les plus réalistes mais également les plus difficiles à mettre en oeuvre. Des exemples de treillis sont montrés au chapitre 2.

5.3 Résultats à valider

La section qui suit explique, sommairement, les possibilités qui devront être explorées, testées, validées, par des travaux futurs.

5.3.1 Processus de précipitation

Le processus inverse à la dissolution est la précipitation. Ce dernier possède lui aussi deux régimes distincts qu'on appellera «régimes de croissance». Ces régimes se distinguent par deux

facteurs qui contrôlent la croissance des cristaux, le taux de croissance étant limité par le plus lent des deux. On qualifiera un régime de «limité par la surface» lorsque l'arrivée d'éléments est rapide et que le processus d'accrochage est lent. Lorsque le processus d'accrochage des éléments est rapide mais que l'arrivée d'éléments est lente, on dira que le régime est «limité par le transport». Tout comme pour le processus de dissolution, ces deux régimes donnent lieu à des fragments de formes distinctives. Le régime limité par la surface, caractérisé par une croissance lente des cristaux, donne des fragments lisses, alors que le régime limité par le transport, caractérisé par une croissance rapide, donne des fragments de morphologie plus complexe. Il reste à démontrer si ces deux régimes de croissance, ainsi que les complexités morphologiques contrastantes décrites ci-haut, sont reliés aux régimes de dissolution de Sahimi et Tsotsis (48).

Un autre phénomène qui nécessiterait une étude plus approfondie est l'atteinte d'un équilibre entre le processus de dissolution et celui de précipitation. Existe-t-il une configuration antagoniste qui permette d'obtenir cet équilibre ? Nous n'en savons rien pour l'instant. Tout ce que nous savons jusqu'à maintenant, c'est qu'inverser la fonction de distribution des probabilités de dissolution et la faire correspondre avec la distribution des probabilités de précipitation ne résulte pas en un système à l'équilibre. De plus, nous savons que la surface lisse des systèmes à croissance lente (régime de précipitation limité par la surface) est liée à un rapport des vitesses de dissolution/précipitation se rapprochant de 1. Sommes-nous capable d'obtenir cette surface lisse avec le simulateur ? Jusqu'à maintenant, aucune expérience en ce sens n'a été concluante. Il faudrait utiliser la même méthode de caractérisation que celle utilisée pour les expériences sur la dissolution, décrites à la première section de ce chapitre. Ce pourrait être une bonne façon de vérifier si les deux régimes de précipitation sont apparentés aux régimes de dissolution de la première section.

5.3.2 Processus de métasomatose (transformation)

Ce processus est celui que nous avons eu le moins le temps de tester. Tout ce que nous avons pu valider, c'est que les images obtenues semblent être relativement fidèles à ce qu'on observe dans la nature. La figure 5.4 montre deux examples de métasomatose.

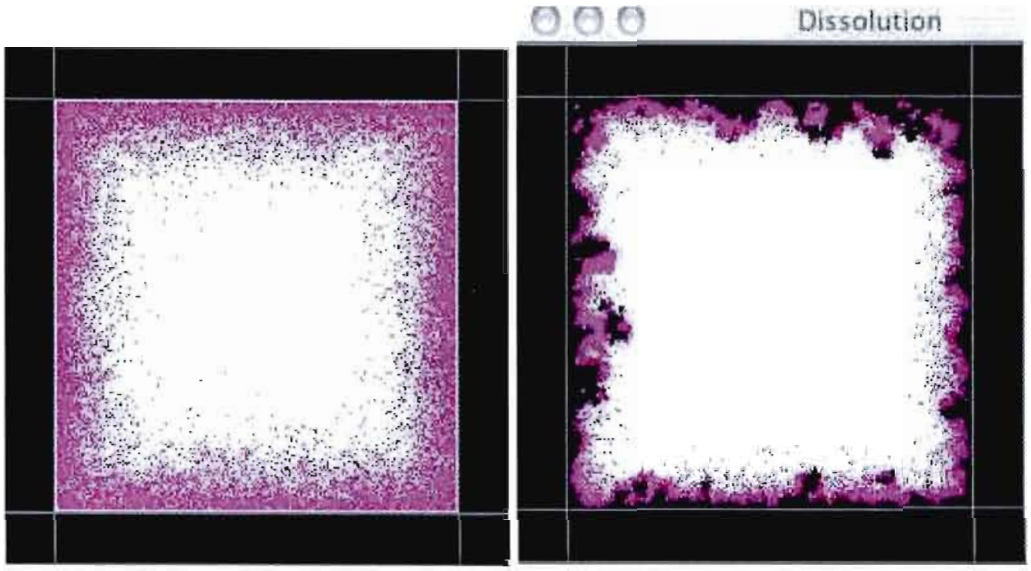


FIG. 5.4 À gauche, le processus de métasomatose isolé, et à droite, les processus de dissolution, précipitation et de métasomatose réunis

Une façon de valider jusqu'à quel point notre processus de métasomatose est fidèle à la réalité consisterait à utiliser des diagrammes isocônes (22). Cette approche, dérivée des techniques de calcul de Gresen (23), est relativement simple. Elle compare la composition minéralogique de deux roches, dans ce cas-ci, la roche «avant» et la roche «après» la simulation.

CONCLUSION

Le projet dont il est question dans ce mémoire consistait à faire la modélisation de brèches hydrothermales et à simuler les processus géologiques qui interviennent dans leur formation et leur altération. Dans ce document, nous avons commencé par introduire certaines notions de géologie. Puis, nous avons décrit deux paradigmes de modélisation, soit les équations aux dérivées partielles et les automates cellulaires. Ces deux paradigmes nous ont été utiles pour modéliser différents processus. Ensuite, nous avons analysé et expliqué le fonctionnement et l'utilisation du simulateur. Finalement, nous avons énuméré et détaillé les tests et les expériences auxquels le simulateur a été soumis. Nous avons également précisé quels processus n'ont pas été testés et validés. Dans le présent chapitre, nous décrirons les résultats obtenus, les problèmes auxquels nous avons dû faire face durant le projet ainsi que les travaux futurs.

Résultats obtenus

Les objectifs généraux du simulateur de brèche étaient de mieux comprendre de quelle façon les brèches hydrothermales se forment et s'altèrent. Plus spécifiquement, le simulateur devait pouvoir montrer qu'il était possible de modéliser le processus de dissolution à l'aide d'un automate cellulaire. En cela, les résultats furent plus que concluants car il a été possible de reprendre et de faire évoluer les réflexions entreprises par d'autres scientifiques (48). Sans le simulateur, il aurait été impossible d'effectuer un aussi grand nombre d'expériences, celles-ci étant coûteuses et dangereuses[2]. Le simulateur de brèches est donc un outil utile de recherche fondamentale qui peut donner des résultats scientifiques concrets.

Problèmes rencontrés

La plus grande difficulté rencontrée fut de mesurer correctement l'ampleur du projet. L'effort nécessaire à la mise en oeuvre de chaque processus était difficile à quantifier au début. Les méthodes classiques de modélisation, c'est-à-dire les méthodes de résolution numérique, se sont avérées être bien plus complexes qu'il paraissait au départ. Il faut dire que l'équation de

[2]On utilise de l'acide fluorhydrique pour dissoudre des quartzites.

diffusion n'est pas représentative de la majorité des équations qu'il faut résoudre en sciences de la terre. L'équation de diffusion est de type parabolique et ce type d'équation est habituellement facile à résoudre en comparaison avec d'autres types (elliptique, hyperbolique). Il n'est donc pas étonnant que certains étudiants de maîtrise soient obligés de se consacrer entièrement à la résolution numérique d'une seule équation.

Au début du projet, le simulateur devait être mis en oeuvre sur un superordinateur de type SGI Origin 3000 et le simulateur devait être parallélisé. Il était tout à fait pertinent de penser à paralléliser notre logiciel puisque nous travaillions sur des processus naturels similaires à ceux qu'on retrouve en climatologie, météorologie, dynamique des fluide, etc. Qui plus est, certaines simulations pouvaient demander des quantités phénoménales de mémoire vive et de temps processeur. Malheureusement, cette idée dû être abandonnée, faute de temps. L'ampleur du travail avait été sous-estimée et un chapitre sur la programmation parallèle que nous avions déjà écrit a donc été omis.

Enfin, un stage de recherche en France à l'été 2004 permit l'amélioration du simulateur et la rédaction d'une partie du mémoire. En particulier, ce stage permit de cerner les limites du paradigme d'automates cellulaires, notamment en ce qui concerne l'apparition des fronts (voir plus loin). En effet, il avait été question de simuler l'apparition de fronts par automates cellulaires mais à force d'essais et de discussions, il est devenu clair que cette approche ne collait vraiment pas. La conclusion est qu'il faut utiliser une méthode de résolution numérique. Le phénomène d'apparition de fronts a donc été classé dans les «travaux futurs».

Travaux futurs

Certains processus ne sont pas mis en oeuvre dans le simulateur. Voici quelques approches possibles qui ont été étudiées dans le cadre de cette maîtrise.

Fragmentation

La mise en œuvre d'un modèle de fragmentation a été documenté dans le cadre de divers travaux (14; 28). Le simulateur pourrait inclure un agent externe similaire à ceux proposés par ces auteurs. Cet agent permettrait de mettre en place une brèche virtuelle dont les fragments pourraient être utilisés comme intrants pour les autres modèles. Un modèle de fragmentation serait tout à fait compatible avec une mise en oeuvre parallèle du simulateur.

Percolation

Quelques modèles de percolation sont disponibles dans la littérature (6). On distingue deux approches pour la mise en oeuvre lorsqu'on travaille sur une matrice : la percolation de «site» (*site percolation*) et la percolation de «liaison» (*bond percolation*). Dans le premier cas, on considère que ce sont les cellules de la matrice qui sont importantes. Dans l'autre cas, on considère plutôt que ce sont les bordures des cellules. La figure 5.5 illustre les deux approches. Conceptuellement, le processus de percolation peut être modélisé par l'approche lattice-gas (18), un automate cellulaire bidimensionnel permettant la résolution des équations Navier-Stokes. L'automate cellulaire peut être construit, soit avec une mise en oeuvre de «site», soit avec une mise en oeuvre «liaison». La méthode lattice-Boltzmann (26; 42), une variante de l'automate cellulaire lattice-gaz, donne de bons résultats en trois dimensions.

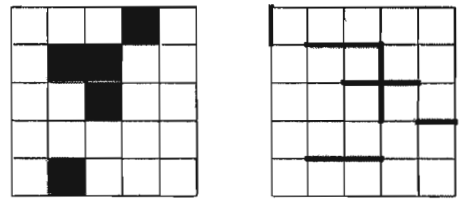


FIG. 5.5 Les deux approches de mise en oeuvre d'un modèle de percolation (46).

Apparition de fronts

Depuis quelques années, de nombreux travaux ont été faits pour modéliser quantitativement les systèmes métasomatiques (24; 9; 36; 40). Les équations différentielles partielles de Korzhinskii (34) (voir chapitre 1) peuvent être résolues numériquement par ordinateur. La distribution des minéraux qui peuvent se développer dans l'espace et dans le temps dû à l'interaction métasomatique entre le fluide et la roche peut donc être prédite. Toutefois, le modèle de Korzhinskii ne tient pas compte dans ses équations des discontinuités de composition définissant les zones métasomatiques. Une révision du modèle mathématique de Korzhinskii permet de tenir compte de ces discontinuités et de l'apparition de fronts (24). Avec ce modèle révisé, des simulations sont possibles (49). Classiquement, on utilise la méthode de Godunov (21) pour les simulations de dynamique des fluides avec discontinuités. Il existe plusieurs algorithmes pour mettre en œuvre cette méthode, la méthode la plus efficace étant la *Piecewise Parabolic Method*

(PPM) de Colella et Woodward (11). Comme pour de nombreuses simulations, la méthode de Godunov sert à résoudre des équations différentielles partielles. L'équation à résoudre en une dimension est la suivante :

$$\frac{\partial Cs}{\partial t} + pv(\frac{\partial Cf}{\partial x}) = 0 \tag{5.1}$$

$$Cf = f(Cs) \tag{5.2}$$

Les variables Cs et Cf représentent respectivement la concentration volumétrique des composants dans le solide et dans le fluide. Cf est habituellement une fonction non linéaire de Cs (équation 5.2). La variable p représente le volume des pores. Les autres variables sont la vitesse (v), le temps (t) et l'espace (x).

Dans le but de pouvoir mettre en oeuvre plus fidèlement le processus de métasomatose (changements de composition) par automates cellulaires, il faut également pouvoir modéliser l'apparition de fronts de concentrations dans notre modèle qui, lui, est en trois dimensions. Les équations 5.3 et 5.4 correspondent à l'équation 5.1 mais en deux et trois dimensions.

$$\frac{\partial Cs}{\partial t} = pv(\frac{\partial Cf}{\partial x} + \frac{\partial Cf}{\partial y}) \tag{5.3}$$

$$\frac{\partial Cs}{\partial t} = pv(\frac{\partial Cf}{\partial x} + \frac{\partial Cf}{\partial y} + \frac{\partial Cf}{\partial z}) \tag{5.4}$$

L'équation à résoudre est une équation de type hyperbolique, équations qui font apparaître des fronts. L'équation hyperbolique la plus connue est l'équation d'onde (la propagation d'une onde sur une corde par exemple). Contrairement à l'équation de diffusion, qui est une équation parabolique, les équations hyperboliques sont particulièrement délicates à résoudre numériquement. Mais la plus grosse contrainte ne vient probablement pas de là. En fait, la difficulté pour intégrer des fronts de diffusion dans notre modèle vient du fait qu'il faut fixer une échelle. En effet, si l'on regarde une roche de suffisamment près, la discontinuité n'en est plus une : le front s'étale. Avec une échelle fixe, déterminée en fonction du processus de fronts, il est à prévoir que les autres processus ne pourront plus interagir comme ils le font maintenant. La raison pour laquelle il est possible d'utiliser le même espace mémoire pour les automates cellulaire et pour

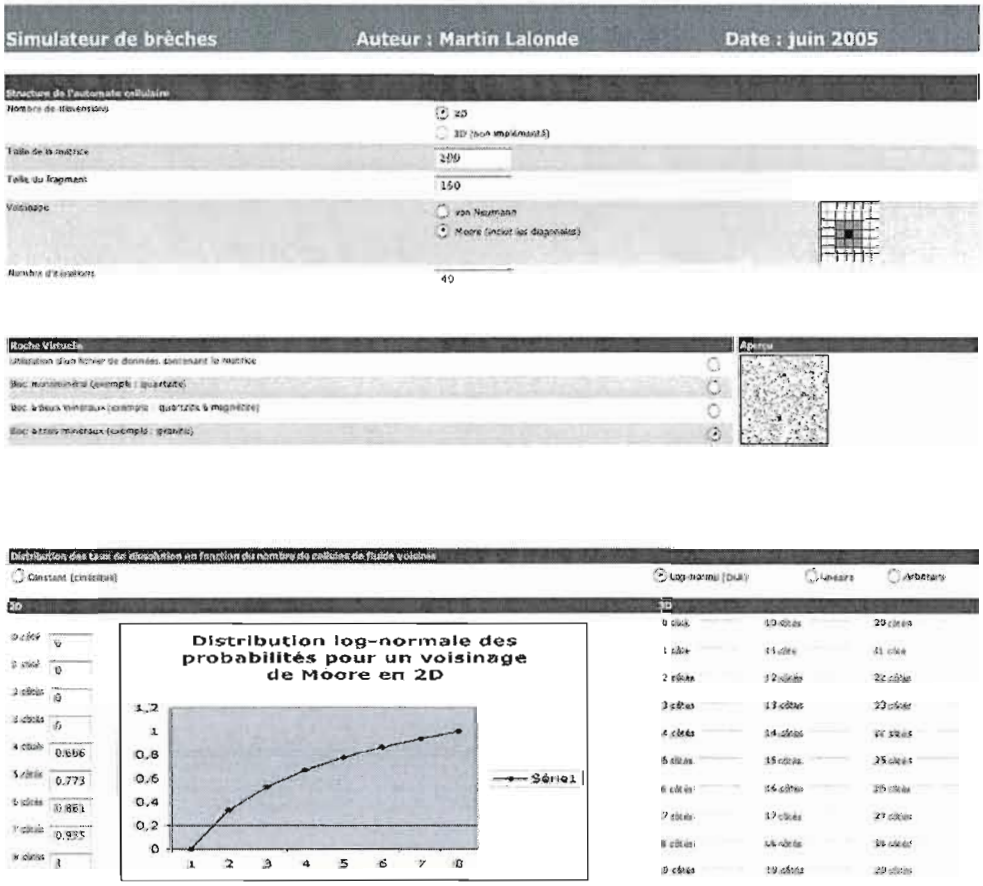
la résolution numérique de l'équation de diffusion est que l'échelle n'est pas fixée pour l'instant et que l'équation de diffusion est facile à résoudre numériquement. Pour modéliser l'apparition de fronts, il faudra séparer les processus basés sur la résolution numérique d'équations des automates cellulaire et les faire communiquer par d'autres moyens. Une solution plus «paresseuse» serait de fixer l'échelle de manière à ne pas voir apparaitre les fronts.

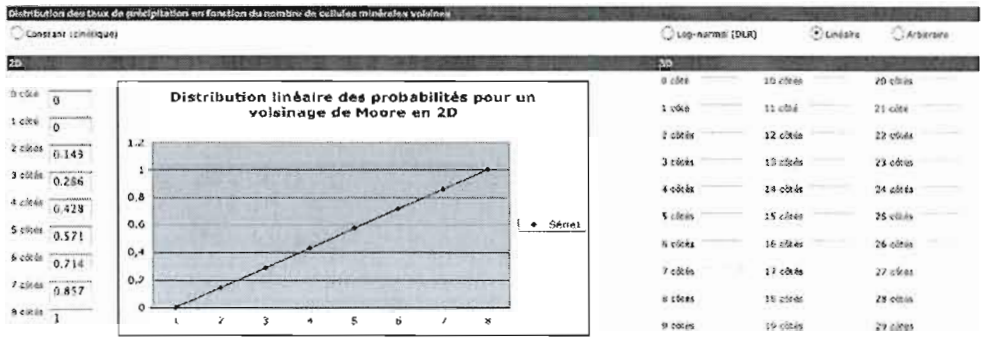
Conclusion

Comme nous l'avons vu, le développement du simulateur s'est achevé avec succès. Des résultats scientifiques tangibles ont été obtenus et un article est en cours de révision pour publication. Le simulateur de brèches, tel qu'il est en ce moment, constituera le point de départ des travaux d'autres étudiants. Un projet possible serait de mettre en oeuvre les processus de fragmentation, percolation et d'apparition de fronts. Une autre possibilité serait de paralléliser le simulateur. Au moment d'écrire ces lignes, un étudiant de géologie est déjà en train d'utiliser l'algorithme de dissolution du simulateur dans ses travaux de classement des brèches. Quoiqu'il en soit, il est intéressant que des chercheurs de deux départements aient pu travailler conjointement à un projet de recherche, à l'encadrement académique et financier d'un étudiant, à la publication d'articles dans des revues scientifiques ainsi qu'à la présentation des résultats dans des conférences. Cette rencontre de deux disciplines fut un grand stimulant intellectuel et m'a permis de m'accomplir et de mieux connaître mes forces en tant que professionnel de l'informatique.

Annexe A

FENÊTRE DE CONFIGURATION DU SIMULATEUR




Distribution des taux de précipitation en fonction du nombre de cellules minérales voisines
Constant (cinétique)
Log-normal (DLR)
Linéaire
Arbitraire
2D
3D
0 côté 0
1 côté 0
2 côtés 0.143
3 côtés 0.286
4 côtés 0.429
5 côtés 0.571
6 côtés 0.714
7 côtés 0.857
8 côtés 1
Distribution linéaire des probabilités pour un voisinage de Moore en 2D
Série1
0 côté
1 côté
2 côtés
3 côtés
4 côtés
5 côtés
6 côtés
7 côtés
8 côtés
9 côtés
10 côtés
11 côté
12 côtés
13 côtés
14 côtés
15 côtés
16 côtés
17 côtés
18 côtés
19 côtés
20 côtés
21 côté
22 côtés
23 côtés
24 côtés
25 côtés
26 côtés
27 côtés
28 côtés
29 côtés


Méthode de diffusion
Jacobi
Gauss-Seidel
Successive Over Relaxation
Calculer Omega optimal
Omega


Options de simulation
Simuler la dissolution
Simuler la précipitation
Simuler les échanges métasomatiques
Simuler la diffusion
Identifier les fragments

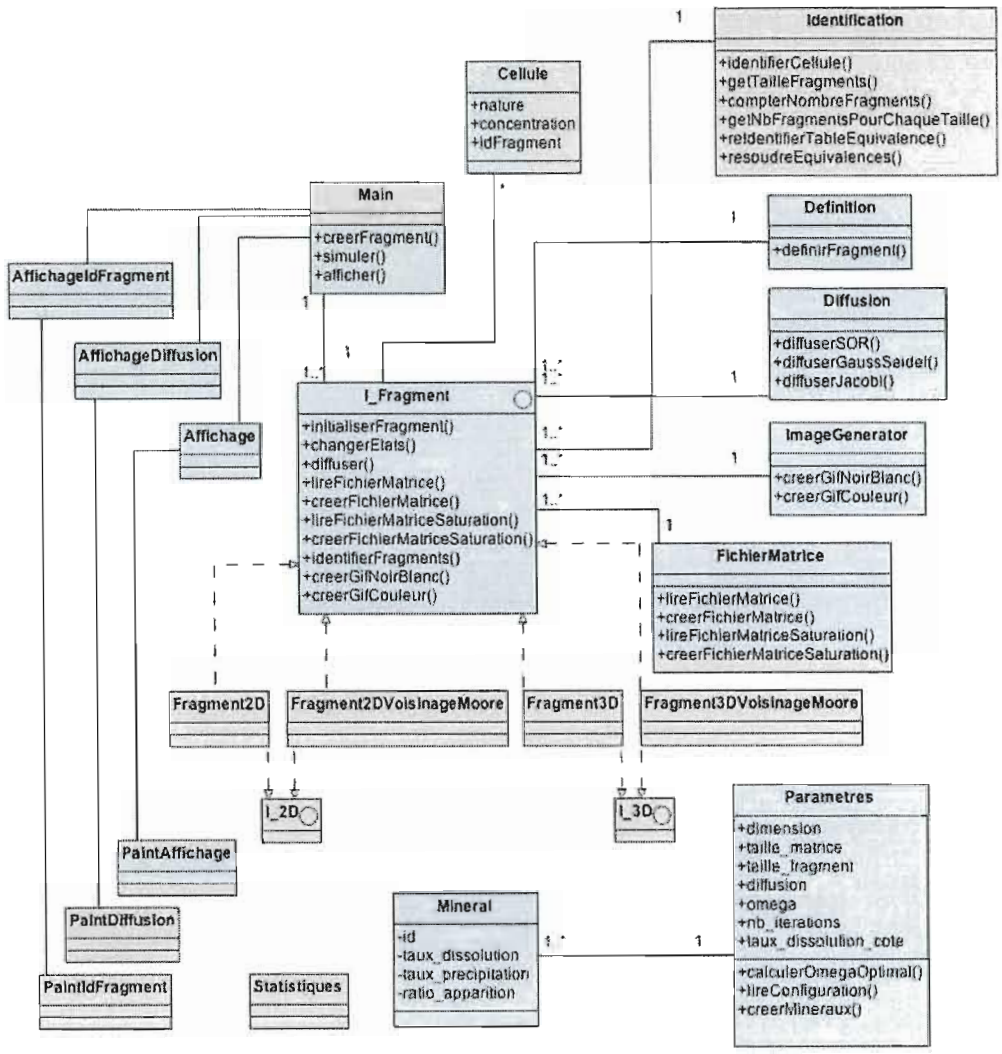

Autres options
Mode débogage
Option d'impression des statistiques dans un fichier
Option de création d'un fichier de la matrice en sortie
Option de création d'un log d'images en noir et blanc après chaque itération
Option de création d'un log d'images en couleur après chaque itération


Minéraux
Nom
Id
Couleur
Id metamorphose
Seuil metamorphose
Taux metamorphose
Taux dissolution
Taux précipitation
Taux apparition
Quartz 1 Blanc 4 (0-100) 15 (0-100) 0.05 (0-1) 1 (0-1) 0.1 (0-1) 0 (0-1)
Feldspath 2 Rose 5 (0-100) 25 (0-100) 0.3 (0-1) 1 (0-1) 1 (0-1) 0 (0-1)
Mica 3 Gris foncé 1 (0-100) 50 (0-100) 0.5 (0-1) 0.5 (0-1) 0.1 (0-1) 0 (0-1)
Minéral transitoire 1 4 Magenta 5 (0-100) 100 (0-100) 0 (0-1) 0.1 (0-1) 0.8 (0-1) 0.5 (0-1)
Minéral transitoire 2 5 Orange 5 (0-100) 30 (0-100) 0.8 (0-1) 0.2 (0-1) 1 (0-1) 1 (0-1)
Porosité 6 Bleu 6 100 0 0 0 0
Fluide 7 Noir 7 100 0 0 0 0
Bordure 8 Noir 8 100 0 0 0 0

Annexe B

DIAGRAMME DE CLASSES

Nous présentons ici le diagramme de classes du simulateur de brèches hydrothermales. Nous présentons uniquement les méthodes publiques pour des raisons de concision. Notons que les classes `Fragment2D`, `Fragment2DVoisinageMoore`, `Fragment3D` et `Fragment3DVoisinageMoore` implémentent toutes les méthodes de l'interface `I_Fragment`. De plus, les méthodes de dissolution, de précipitation et de transformation (échanges métasomatiques) se trouvent dans les quatre classes implémentant l'interface `I_Fragment` et sont privées. Elles sont appelées par la méthode publique `changerEtat()`. Finalement, on remarque que les interfaces I_2D et I_3D sont vides. En fait, le seul rôle de ces deux interfaces est d'obliger la redéfinition d'une méthode `getMatrice()` qui retourne une matrice bidimensionnelle dans le premier cas et tridimensionelle dans le second cas.


Identification
+identifierCellule()
+getTailleFragments()
+compterNombreFragments()
+getNbFragmentsPourChaqueTaille()
+reIdentifierTableEquivalence()
+resoudreEquivalences()
Cellule
+nature
+concentration
+idFragment
Main
+creerFragment()
+simuler()
+afficher()
Definition
+definirFragment()
AffichageIdFragment
AffichageDiffusion
Diffusion
+diffuserSOR()
+diffuserGaussSeidel()
+diffuserJacobi()
I_Fragment
+initialiserFragment()
+changerEtats()
+diffuser()
+lireFichierMatrice()
+creerFichierMatrice()
+lireFichierMatriceSaturation()
+creerFichierMatriceSaturation()
+identifierFragments()
+creerGifNoirBlanc()
+creerGifCouleur()
Affichage
ImageGenerator
+creerGifNoirBlanc()
+creerGifCouleur()
FichierMatrice
+lireFichierMatrice()
+creerFichierMatrice()
+lireFichierMatriceSaturation()
+creerFichierMatriceSaturation()
Fragment2D
Fragment2DVoisinageMoore
Fragment3D
Fragment3DVoisinageMoore
I_2D
I_3D
Parametres
+dimension
+taille_matrice
+taille_fragment
+diffusion
+omega
+nb_iterations
+taux_dissolution_cote
+calculerOmegaOptimal()
+lireConfiguration()
+creerMineraux()
PaintAffichage
PaintDiffusion
PaintIdFragment
Statistiques
Mineral
-id
-taux_dissolution
-taux_precipitation
-ratio_apparition

Annexe C

EXTRAITS DE CODE DU SIMULATEUR DE BRÈCHES

C.1 Classe Fragment2D

```
package simulateur_13_avril_2005;

import java.io.File;
import java.io.OutputStreamWriter;
import java.io.FileOutputStream;
import java.io.Writer;
import java.io.BufferedWriter;
import java.io.BufferedReader;
import java.util.StringTokenizer;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.Properties;
import java.util.LinkedList;
import java.util.TreeMap;

/**
 * <p>Title: Simulateur de breche</p>
 *
 * <p>Description: </p>
 *
 * <p>Copyright: Copyright (c) 2005</p>
 *
 * <p>Company: UQAM</p>
 *
 * @author Martin Lalonde
 * @version 1.0
 */

public class Fragment2D implements Fragment, I_2D {

    // listeIndices[i][0] = id
    // listeIndices[i][1] = indice_inferieur
    // listeIndices[i][2] = indice_superieur
    private double[][] listeIndices = new double[Parametres.NB_MAX_MINERAUX][3];
```

```
public Cellule matrice[][] = new Cellule[Parametres.TAILLE_MATRICE][
                             Parametres.TAILLE_MATRICE];
public Cellule matrice_temp[][] = new Cellule[Parametres.TAILLE_MATRICE][
                                  Parametres.TAILLE_MATRICE];


public Fragment2D() {

    for (int i = 0; i < Parametres.TAILLE_MATRICE; i++) {
        for (int j = 0; j < Parametres.TAILLE_MATRICE; j++) {
            matrice[i][j] = new Cellule();
            matrice_temp[i][j] = new Cellule();
        }
    }

    determinerListeIndicesApparition();
}


public Cellule[][] getMatrice() {

    return matrice;
}

public Cellule[][] getMatriceTemp() {
    return matrice_temp;
}


public void lireFichierMatrice() {
    FichierMatrice.lireFichierMatrice(this);
}

public void lireFichierMatriceSaturation() {
    FichierMatrice.lireFichierMatriceSaturation(this);
}

public void creerFichierMatrice(long time) {
    FichierMatrice.creerFichierMatrice(this, time);
}

public void creerFichierMatriceSaturation(long time) {
    FichierMatrice.creerFichierMatriceSaturation(this, time);
}


public void identifierFragments(Identification uneIdentification) {

    // Utilisation d'un voisinage de type "4-connected"
    // Deux cellules reliés par la diagonale ne sont donc pas connectés

    for (int i = 1; i < Parametres.TAILLE_MATRICE - 1; i++) {
```

```
        for (int j = 1; j < Parametres.TAILLE_MATRICE - 1; j++) {
            uneIdentification.identifierCellule(this, i, j);
        }
    }

    uneIdentification.reIdentifierTableEquivalence();

    // Passe pour remplacer les identifiants temporaires et resoudre les equivalences
    for (int i = 1; i < Parametres.TAILLE_MATRICE; i++) {
        for (int j = 1; j < Parametres.TAILLE_MATRICE; j++) {
            // Ce if est necessaire pour faire apparaitre les idFragments qui ne
            // sont pas dans la table d'equivalence
            uneIdentification.resoudreEquivalences(this, i, j);

        }
    }

}


public void creerGifNoirBlanc(int step, long time) {
    ImageGenerator.creerImagePourNIH(step, this, time);
}

public void creerGifCouleur(int step, long time) {
    ImageGenerator.creerImageLog(step, this, time);
}

private double getParametreAlpha(int _nombreVoisins) {
    switch (_nombreVoisins) {
    case 0:
        return Parametres.TAUX_DISSOLUTION_COTE[0];

    case 1:
        return Parametres.TAUX_DISSOLUTION_COTE[1];

    case 2:
        return Parametres.TAUX_DISSOLUTION_COTE[2];

    case 3:
        return Parametres.TAUX_DISSOLUTION_COTE[3];

    case 4:
        return Parametres.TAUX_DISSOLUTION_COTE[4];
    default:
        System.out.println(
                "Erreur avec la methode getParametreAlpha");
        break;
    }
    return 0;
}

private double getParametreBeta(int _nombreVoisins) {
    switch (_nombreVoisins) {
```

```
    case 0:
        return Parametres.TAUX_PRECIPITATION_COTE[0];

    case 1:
        return Parametres.TAUX_PRECIPITATION_COTE[1];

    case 2:
        return Parametres.TAUX_PRECIPITATION_COTE[2];

    case 3:
        return Parametres.TAUX_PRECIPITATION_COTE[3];

    case 4:
        return Parametres.TAUX_PRECIPITATION_COTE[4];
    default:
        System.out.println(
                "Erreur avec la methode getParametreBeta");
        break;
    }
    return 0;
}


public void initialiserFragment() {

    // Depuis un fichier source
    // Par exemple pour faire une seconde phase de diffusion-transformation
    if (Parametres.TYPE_ROCHE == Parametres.UTILISER_MATRICE_INPUT) {
        lireFichierMatrice();
        lireFichierMatriceSaturation();
    } else {

        Definition.definirFragment(this);
    }

}


private void remplacerMatrice() {

    // On remplace la matrice originale par les nouvelles valeurs (temporaires)
    for (int i = 1; i < Parametres.TAILLE_MATRICE - 1; i++) {
        for (int j = 1; j < Parametres.TAILLE_MATRICE - 1; j++) {
            matrice[i][j].nature = matrice_temp[i][j].nature;
            matrice[i][j].saturation = matrice_temp[i][j].saturation;
            matrice[i][j].idFragment = matrice_temp[i][j].idFragment;
        }
    }
}

public void diffuser() {

    if (Parametres.DIFFUSION == Parametres.JACOBI) {
        Diffusion.diffuserJacobi(this);
```

```
    } else if (Parametres.DIFFUSION == Parametres.GAUSS_SEIDEL) {
        Diffusion.diffuserGaussSeidel(this);
    } else if (Parametres.DIFFUSION ==
                Parametres.SUCCESSIVE_OVER_RELAXATION) {
        Diffusion.diffuserSuccessiveOverRelaxation(this);
    }

}


private int nombreVoisinsFluide(int i, int j) {
    int count = 0;

    if (matrice[i - 1][j].nature == Parametres.FLUIDE ||
        matrice[i - 1][j].nature == Parametres.BORDURE) {
        count++;
    }
    if (matrice[i + 1][j].nature == Parametres.FLUIDE ||
        matrice[i + 1][j].nature == Parametres.BORDURE) {
        count++;
    }
    if (matrice[i][j - 1].nature == Parametres.FLUIDE ||
        matrice[i][j - 1].nature == Parametres.BORDURE) {
        count++;
    }
    if (matrice[i][j + 1].nature == Parametres.FLUIDE ||
        matrice[i][j + 1].nature == Parametres.BORDURE) {
        count++;
    }

    return count;

}

private int nombreVoisinsSolide(int i, int j) {
    int count = 0;

    if (matrice[i - 1][j].nature != Parametres.FLUIDE &&
        matrice[i - 1][j].nature != Parametres.BORDURE) {
        count++;
    }
    if (matrice[i + 1][j].nature != Parametres.FLUIDE &&
        matrice[i + 1][j].nature != Parametres.BORDURE) {
        count++;
    }
    if (matrice[i][j - 1].nature != Parametres.FLUIDE &&
        matrice[i][j - 1].nature != Parametres.BORDURE) {
        count++;
    }
    if (matrice[i][j + 1].nature != Parametres.FLUIDE &&
        matrice[i][j + 1].nature != Parametres.BORDURE) {
        count++;
    }
```

```
    return count;

}


public void changerEtats() {

    /*
      Dissoudre, précipiter, transformer, etc
     */
    boolean estDissout = false;

    for (int i = 1; i < Parametres.TAILLE_MATRICE - 1; i++) {
        for (int j = 1; j < Parametres.TAILLE_MATRICE - 1; j++) {
            if (matrice[i][j].nature != Parametres.FLUIDE &&
                matrice[i][j].nature != Parametres.BORDURE) {
                if (Parametres.DISSOUDRE) {
                    estDissout = dissoudre(i, j);
                }
                if (!estDissout) {
                    if (Parametres.TRANSFORMER) {
                        transformer(i, j);
                    }
                }
            } else {
                if (Parametres.PRECIPITER) {
                    precipiter(i, j);
                }
            }
        }
    }

    remplacerMatrice();
}

private void transformer(int i, int j) {
    double aleatoire = 0;
    double taux_metamorphose = 0;
    Mineral unMineral;

    unMineral = (Mineral) Parametres.listeMineraux.get((new Integer(matrice[
            i][j].nature)));

    if (matrice[i][j].saturation > unMineral.getSeuilMetamorphose()) {

        // Il faudra verifier le type de mineral a dissoudre
        taux_metamorphose = matrice[i][j].saturation /
                            Parametres.CONSTANTE_METAMORPHOSE;

        // Determiner si l'on dissout
        aleatoire = Math.random();

        if (aleatoire < taux_metamorphose) {
            matrice_temp[i][j].nature = unMineral.getIdMetamorphose();
```

```
            matrice_temp[i][j].saturation = matrice[i][j].saturation;
            Statistiques.nbCellulesTransformees++;
        }
    }
}

private void determinerListeIndicesApparition() {

    /* Methode appelee dans le constructeur */

    // Initialiser le tableau d'indices
    for (int i = 0; i < Parametres.NB_MAX_MINERAUX; i++) {
        for (int j = 0; j < 3; j++) {
            listeIndices[i][j] = 0;
        }
    }

    java.util.Iterator i;

    java.util.Collection listeMineraux = Parametres.listeMineraux.values();
    i = listeMineraux.iterator();

    Mineral unMineral = null;
    int j = 1;
    while (i.hasNext()) {
        unMineral = (Mineral) i.next();
        if (unMineral.getRatioApparition() != 0) {
            // listeIndices[j][0] = id mineral
            // listeIndices[j][1] = indice_inferieur
            // listeIndices[j][2] = indice_superieur

            listeIndices[j][0] = unMineral.getId();
            listeIndices[j][1] = listeIndices[j - 1][2];
            listeIndices[j][2] = listeIndices[j][1] +
                                 unMineral.getRatioApparition();
            j++;
        }

    }

}

private Mineral determinerMineralPourPrecipiter() {

    double aleatoire = 0;
    Mineral unMineral = null;

    aleatoire = Math.random();

    boolean estEntreDeuxIndices = false;
    int i = 0;
    while (!estEntreDeuxIndices) {
        if (listeIndices[i][1] < aleatoire &&
            listeIndices[i][2] > aleatoire) {
```

```
            estEntreDeuxIndices = true;
            unMineral = (Mineral) Parametres.listeMineraux.get(new Integer((int)
                    listeIndices[i][0]));
        }
        i++;
    }

    return unMineral;
}

private void precipiter(int i, int j) {

    double aleatoire = 0;
    double indice_precipitation = 0;
    Mineral unMineral;

    unMineral = determinerMineralPourPrecipiter();

    // Il faudra verifier le type de mineral a dissoudre
    indice_precipitation = unMineral.getTauxPrecipitation() *
                           getParametreBeta(nombreVoisinsSolide(i, j));

    // Determiner si l'on dissout
    aleatoire = Math.random();

    if (aleatoire < indice_precipitation) {

        matrice_temp[i][j].nature = unMineral.getId();
        matrice_temp[i][j].saturation = 100;
        Statistiques.nbCellulesPrecipites++;
    } else {
        matrice_temp[i][j].nature = matrice[i][j].nature;
        matrice_temp[i][j].saturation = matrice[i][j].saturation;
    }

}

private boolean dissoudre(int i, int j) {

    double aleatoire = 0;
    double indice_dissolution = 0;
    Mineral unMineral;
    boolean estDissout = false;

    unMineral = (Mineral) Parametres.listeMineraux.get((new Integer(matrice[
            i][j].nature)));

    if (unMineral.getId() == Parametres.POROSITE) {
        if (nombreVoisinsFluide(i, j) >= 1) {
            matrice_temp[i][j].nature = Parametres.FLUIDE;
            matrice_temp[i][j].saturation = 100;
            estDissout = true;
        } else {
            matrice_temp[i][j].nature = matrice[i][j].nature;
```

```
                matrice_temp[i][j].saturation = matrice[i][j].saturation;
            }

        } else {

            indice_dissolution = unMineral.getTauxDissolution() *
                                 getParametreAlpha(nombreVoisinsFluide(i, j));

            // Determiner si l'on dissout
            aleatoire = Math.random();

            if (aleatoire < indice_dissolution) {
                matrice_temp[i][j].nature = Parametres.FLUIDE;
                matrice_temp[i][j].saturation = 100;
                estDissout = true;
                Statistiques.nbCellulesDissoutes++;

            } else {
                matrice_temp[i][j].nature = matrice[i][j].nature;
                matrice_temp[i][j].saturation = matrice[i][j].saturation;
            }
        }

        return estDissout;

    }
}
```

C.2 Classe Diffusion

```
package simulateur_13_avril_2005;

/**
 * <p>Title: Simulateur de breche</p>
 *
 * <p>Description: </p>
 *
 * <p>Copyright: Copyright (c) 2005</p>
 *
 * <p>Company: UQAM</p>
 *
 * @author Martin Lalonde
 * @version 1.0
 */
public class Diffusion {
    public Diffusion() {
    }


    public static void diffuserSuccessiveOverRelaxation(I_2D unFragment) {
```

```
        // Algorithme pris dans Hansen, 1995, avec ordonnancement par parite
        // pour faciliter la mise en oeuvre parallele

        int k;
        int j;

        // On commence apres la bordure et on termine avant
        for (int b = 0; b <= 1; b++) {
            for (int i = 0; i < Parametres.TAILLE_MATRICE; i++) {

                k = (i + b) % 2;
                j = 2 - k;

                while (j < (Parametres.TAILLE_MATRICE - k)) {
                    if (unFragment.getMatrice()[i][j].nature !=
                        Parametres.BORDURE &&
                        unFragment.getMatrice()[i][j].nature !=
                        Parametres.FLUIDE) {

                        unFragment.getMatrice()[i][j].saturation =
                                prochaineRelaxation(unFragment.getMatrice()[i][
                                j].saturation,
                                unFragment.getMatrice()[i - 1][j].saturation,
                                unFragment.getMatrice()[i + 1][j].saturation,
                                unFragment.getMatrice()[i][j + 1].saturation,
                                unFragment.getMatrice()[i][j - 1].saturation);

                    }
                    j = j + 2;
                }
            }
        }
    }


    private static double prochaineRelaxation(double c, double w, double e,
                                              double n, double s) {

        double res;

        res = (w + e + n + s) / 4 - c;

        return (c + Parametres.OMEGA * res);

    }

    public static void diffuserGaussSeidel(I_2D unFragment) {
// Cette methode converge 2 fois plus vite que Jacobi car on n'utilise pas de matrice temporaire

        // On commence apres la bordure et on termine avant
        for (int i = 1; i < Parametres.TAILLE_MATRICE - 1; i++) {
            for (int j = 1; j < Parametres.TAILLE_MATRICE - 1; j++) {
                if (unFragment.getMatrice()[i][j].nature != Parametres.BORDURE &&
                    unFragment.getMatrice()[i][j].nature != Parametres.FLUIDE) {
```

```
                    unFragment.getMatrice()[i][j].saturation =
                            (unFragment.getMatrice()[i - 1][j].saturation
                             + unFragment.getMatrice()[i + 1][j].saturation
                             + unFragment.getMatrice()[i][j - 1].saturation
                             + unFragment.getMatrice()[i][j + 1].saturation) /
                            4;
                }
            }
        }
    }

    public static void diffuserJacobi(I_2D unFragment) {

// On commence apres la bordure et on termine avant
        for (int i = 1; i < Parametres.TAILLE_MATRICE - 1; i++) {
            for (int j = 1; j < Parametres.TAILLE_MATRICE - 1; j++) {
                if (unFragment.getMatrice()[i][j].nature != Parametres.BORDURE &&
                    unFragment.getMatrice()[i][j].nature != Parametres.FLUIDE) {

                    unFragment.getMatriceTemp()[i][j].saturation =
                            (unFragment.getMatrice()[i - 1][j].saturation
                             + unFragment.getMatrice()[i + 1][j].saturation
                             + unFragment.getMatrice()[i][j - 1].saturation
                             + unFragment.getMatrice()[i][j + 1].saturation) /
                            4;

                    unFragment.getMatriceTemp()[i][j].nature = unFragment.
                            getMatrice()[i][j].nature;
                }
            }
        }

        remplacerMatrice(unFragment);

    }

    private static void remplacerMatrice(I_2D unFragment) {

        // On remplace la matrice originale par les nouvelles valeurs (temporaires)
        for (int i = 1; i < Parametres.TAILLE_MATRICE - 1; i++) {
            for (int j = 1; j < Parametres.TAILLE_MATRICE - 1; j++) {
                unFragment.getMatrice()[i][j].nature = unFragment.
                        getMatriceTemp()[i][j].nature;
                unFragment.getMatrice()[i][j].saturation = unFragment.
                        getMatriceTemp()[i][j].saturation;
                unFragment.getMatrice()[i][j].idFragment = unFragment.
                        getMatriceTemp()[i][j].idFragment;
            }
        }
    }

}
```

C.3 Classe Identification

```
package simulateur_13_avril_2005;

import java.util.TreeMap;

/**
 * <p>Title: Simulateur de breche</p>
 *
 * <p>Description: </p>
 *
 * <p>Copyright: Copyright (c) 2005</p>
 *
 * <p>Company: UQAM</p>
 *
 * @author Martin Lalonde
 * @version 1.0
 */
public class Identification {

    public Identification() {
        idFragment = 1;

        for (int i = 0; i < TAILLE_TABLE_EQUIVALENCE; i++) {
            tailleFragments[i] = 0;
        }

        for (int i = 0; i < TAILLE_TABLE_EQUIVALENCE; i++) {
            table_equivalence[i] = new TreeMap();
        }

    }


    // Constantes
    private final int AUCUN_VOISIN_IDENTIFIE = 0;
    private final int UN_VOISIN_IDENTIFIE = 1;
    private final int DEUX_VOISINS_IDENTIFIES = 2;

    public static final int TAILLE_TABLE_EQUIVALENCE = 20 *
            Parametres.TAILLE_FRAGMENT;

    private int tailleFragments[] = new int[TAILLE_TABLE_EQUIVALENCE];
    private TreeMap nbFragmentsTaille = new TreeMap();

    // Les tailles sont a revoir (surtout pour le 3D)
    private TreeMap table_equivalence[] = new TreeMap[TAILLE_TABLE_EQUIVALENCE];
    private int table_equivalence_resolue[] = new int[TAILLE_TABLE_EQUIVALENCE];
```

```
private int idFragment = 0;


public int[] getTailleFragments(I_2D unFragment) {
    /* Conserve le nombre de cellules associé à chaque fragment distinct
       incluant le fluide
     */

    for (int i = 0; i < Parametres.TAILLE_MATRICE; i++) {
        for (int j = 0; j < Parametres.TAILLE_MATRICE; j++) {
            tailleFragments[unFragment.getMatrice()[i][j].idFragment]++;
        }
    }

    return tailleFragments;

}

public int compterNbFragments() {
    /* Retourne le nombre de idFragment differents
       donc le vrai nombre de fragments distincts
       incluant le idFragment du fluide
     */

    int count = 0;
    for (int i = 1; i < TAILLE_TABLE_EQUIVALENCE; i++) {
        if (tailleFragments[i] > 0) {
            count++;
        }
    }
    return count;

}

public TreeMap getNbFragmentsPourChaqueTaille() {

    Integer nbFragments;
    for (int i = 0; i < TAILLE_TABLE_EQUIVALENCE; i++) {
        int nbFragmentsTemp = 0;
        if (tailleFragments[i] != 0) {
            nbFragments = (Integer) nbFragmentsTaille.get(new Integer(
                    tailleFragments[i]));
            if (nbFragments == null) {
                nbFragmentsTemp++;
            } else {
                nbFragmentsTemp = nbFragments.intValue() + 1;
            }
            nbFragmentsTaille.put(new Integer(tailleFragments[i]),
                                  new Integer(nbFragmentsTemp));
        }
    }

    return nbFragmentsTaille;
```

```
}

private int nbVoisinsDejaParcourusConnectes(I_2D unFragment, int i, int j) {

    int count = 0;

    if (unFragment.getMatrice()[i - 1][j].nature != Parametres.FLUIDE) {
        count++;
    }
    if (unFragment.getMatrice()[i][j - 1].nature != Parametres.FLUIDE) {
        count++;
    }

    return count;
}

private int getIdentifiantCelluleConnectee(I_2D unFragment, int i, int j) {

    int idFragment1 = Integer.MAX_VALUE;
    int idFragment2 = Integer.MAX_VALUE;

    if (unFragment.getMatrice()[i - 1][j].nature != Parametres.FLUIDE) {

        idFragment1 = unFragment.getMatrice()[i - 1][j].idFragment;
    }

    if (unFragment.getMatrice()[i][j - 1].nature != Parametres.FLUIDE) {
        idFragment2 = unFragment.getMatrice()[i][j - 1].idFragment;
    }

    return (idFragment1 < idFragment2 ? idFragment1 : idFragment2);

}

private void noterEquivalence(int idFragment1, int idFragment2) {

    // idFragment1 = i-1    et idFragment2 = j-1

    if (idFragment1 == idFragment2) {
        table_equivalence[idFragment1].put(new Integer(idFragment1), null);
    } else if (idFragment1 < idFragment2) {
        table_equivalence[idFragment2].put(new Integer(idFragment1), null);
    } else if (idFragment1 > idFragment2) {
        table_equivalence[idFragment1].put(new Integer(idFragment2), null);
    }

}

public void identifierCellule(I_2D unFragment, int i, int j) {

    if (unFragment.getMatrice()[i][j].nature == Parametres.FLUIDE) {
        unFragment.getMatrice()[i][j].idFragment = 0;
    } else {
```

```
        if (nbVoisinsDejaParcourusConnectes(unFragment, i, j) ==
            AUCUN_VOISIN_IDENTIFIE) {
            // Attribuer un nouvel identifiant unique a la cellule
            unFragment.getMatrice()[i][j].idFragment = idFragment++;
            /*
                        System.out.print("AUCUN_VOISIN_IDENTIFIE : ");
             System.out.println(unFragment.getMatrice()[i][j].idFragment);
             */
        } else
        if (nbVoisinsDejaParcourusConnectes(unFragment, i, j) ==
            UN_VOISIN_IDENTIFIE) {
            /*
                        System.out.print("UN_VOISIN_IDENTIFIE : ");
             System.out.println(unFragment.getMatrice()[i][j].idFragment);
             */

            unFragment.getMatrice()[i][j].idFragment =
                    getIdentifiantCelluleConnectee(unFragment, i, j);

        } else
        if (nbVoisinsDejaParcourusConnectes(unFragment, i, j) ==
            DEUX_VOISINS_IDENTIFIES) {
            unFragment.getMatrice()[i][j].idFragment =
                    getIdentifiantCelluleConnectee(unFragment, i, j);

            // Si l'identifiant est different, on note l'equivalence
            noterEquivalence(unFragment.getMatrice()[i - 1][j].idFragment,
                             unFragment.getMatrice()[i][j - 1].idFragment);

            /*
                        System.out.print("DEUX_VOISINS_IDENTIFIES : ");
             System.out.println(unFragment.getMatrice()[i][j].idFragment);
             */
        }
    }
}

public void reIdentifierTableEquivalence() {

    if (Parametres.DEBUG) {
        System.out.print("Table equivalence : ");
        for (int i = 0; i < Parametres.TAILLE_MATRICE; i++) {
            System.out.print(i);
            System.out.print(table_equivalence[i]);
            System.out.print(" ");
        }
        System.out.println("");
    }

    int plusPetiteCle;
    java.util.Collection uneCollection;
    java.util.Iterator unIterateur;
    for (int i = 0; i < TAILLE_TABLE_EQUIVALENCE; i++) {
        if (!table_equivalence[i].isEmpty()) {
```

```
            plusPetiteCle = ((Integer) table_equivalence[i].firstKey()).
                            intValue();

            uneCollection = table_equivalence[i].keySet();

            unIterateur = uneCollection.iterator();

            while (unIterateur.hasNext()) {
                Integer entier = (Integer) unIterateur.next();
                //if(entier.intValue() != plusPetiteCle || entier.intValue() != i) {
                table_equivalence[entier.intValue()].put(new Integer(
                        plusPetiteCle), null);

                //}
            }

        }

    }

    if (Parametres.DEBUG) {
        System.out.print("Table equivalence : ");
        for (int i = 0; i < Parametres.TAILLE_MATRICE; i++) {
            System.out.print(i);
            System.out.print(table_equivalence[i]);
            System.out.print(" ");
        }
        System.out.println("");
    }

    for (int i = 0; i < TAILLE_TABLE_EQUIVALENCE; i++) {
        if (!table_equivalence[i].isEmpty()) {

            plusPetiteCle = ((Integer) table_equivalence[i].firstKey()).
                            intValue();

            table_equivalence_resolue[i] = plusPetiteCle;

        } else {
            table_equivalence_resolue[i] = 0;
        }
    }

    for (int i = 0; i < TAILLE_TABLE_EQUIVALENCE; i++) {
        if (table_equivalence_resolue[i] != 0) {
            table_equivalence_resolue[i] = getEquivalence(i,
                    table_equivalence_resolue[i]);
        }
    }

}

private int getEquivalence(int i, int v) {
```

```
        if (i == v) {
            return i;
        } else if (v < i) {
            table_equivalence_resolue[i] = getEquivalence(v,
                    table_equivalence_resolue[v]);
        }

        return table_equivalence_resolue[i];
    }


    public void resoudreEquivalences(I_2D unFragment, int i, int j) {

        if (table_equivalence_resolue[unFragment.getMatrice()[i][j].idFragment] !=
            0) {
            unFragment.getMatrice()[i][j].idFragment =
                    table_equivalence_resolue[unFragment.getMatrice()[i][j].
                    idFragment];
        }

    }


}
```

C.4 Classe Main

```
package simulateur_13_avril_2005;

import javax.swing.JFrame;
import java.util.Calendar;
import java.io.File;
import java.io.IOException;
import java.nio.channels.FileChannel;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.nio.MappedByteBuffer;

/**
 * <p>Title: Simulateur de breche</p>
 *
 * <p>Description: </p>
 *
 * <p>Copyright: Copyright (c) 2005</p>
 *
 * <p>Company: UQAM</p>
 *
 * @author Martin Lalonde
 * @version 1.0
```

```
 */

public class Main {


    public Main() {
    }

    public static void main(String[] args) {

        Fragment fragmentInitial;

        Main main = new Main();

        // Définir un identifiant unique pour la création des fichiers de sortie
        Calendar unCalendrier = Calendar.getInstance();
        long time = unCalendrier.getTimeInMillis();

        File image_dir = new File("output" + time);
        image_dir.mkdir();

        // Copier le fichier de configuration dans le répertoire de sortie
        try {
            copy(new File(Parametres.NOM_FICHIER_CONFIGURATION),
                 new File("output" + time + "/" +
                          Parametres.NOM_FICHIER_CONFIGURATION));
        } catch (IOException e) {
            System.out.println("Impossible de copier le fichier de
             configuration dans le dossier de la simulation");
        }

        Parametres.creerMineraux();

        if (Parametres.UTILISER_FICHIER_CONFIG) {
            Parametres.lireConfiguration();
        }

        Statistiques moduleStatistiques = new Statistiques();

        fragmentInitial = main.creerFragment();
        //fragment2 = main.creerFragment();

        moduleStatistiques.calculerNbTotalCellules();
        moduleStatistiques.calculerNbCellulesDepart((I_2D) fragmentInitial);

        // Pour simuler l'alteration de plusieurs fragments en parallele,
        // inserer ici des threads et rappeler N fois la methode creerRoche(___)
        // et N fois simuler(____)

        main.simuler(fragmentInitial, time);
        //main.simuler(fragment2);

        moduleStatistiques.calculerNbCellulesFin((I_2D) fragmentInitial);
```

```
    // Pour initialiser les variables dans le constructeur
    if (Parametres.IDENTIFIER) {
        Identification uneIdentification = new Identification();
        fragmentInitial.identifierFragments(uneIdentification);

        moduleStatistiques.calculerTaillesFragments(uneIdentification,
                (I_2D) fragmentInitial);
        moduleStatistiques.calculerNbFragments(uneIdentification);
        moduleStatistiques.calculerNbFragmentsChaqueTaille(
                uneIdentification);
    }

    main.afficher(fragmentInitial, 1);
    //main.afficher(fragment2, 2);

    if (Parametres.UTILISER_MATRICE_OUTPUT) {

        fragmentInitial.creerFichierMatrice(time);
        fragmentInitial.creerFichierMatriceSaturation(time);
    }

    if (Parametres.PRINT_STATISTIQUES) {
        moduleStatistiques.ecrireStatistiquesFichier(time);
    } else {
        moduleStatistiques.ecrireStatistiquesEcran();
    }

    System.out.println("Fin normale du programme");

}


public Fragment creerFragment() {

    Fragment unFragment;

    if (Parametres.DIMENSION == Parametres._2D) {
        unFragment = new Fragment2D();
    } else if (Parametres.DIMENSION == Parametres._3D) {
        unFragment = new Fragment3D();
    } else if (Parametres.DIMENSION == Parametres._2D_MOORE) {
        unFragment = new Fragment2DVoisinageMoore();
    } else {
        unFragment = null;
    }

    unFragment.initialiserFragment();

    return unFragment;
}


public void afficher(Fragment unFragment, int noFragmentInitial) {
```

```
    if (Parametres.DIMENSION == Parametres._2D) {
        JFrame f = new Affichage((I_2D) unFragment, "Minéraux");
        f.setLocation(0, 0);
        f.show();

        JFrame f2 = new AffichageDiffusion((I_2D) unFragment, "Diffusion");
        f2.setLocation(200, 0);
        f2.show();

        JFrame f3 = new AffichageIdFragment((I_2D) unFragment,
                                            noFragmentInitial);
        f3.setLocation(400, 0);
        f3.show();

    } else if (Parametres.DIMENSION == Parametres._2D_MOORE) {
        JFrame f = new Affichage((I_2D) unFragment, "Minéraux");
        f.setLocation(0, 0);
        f.show();

        JFrame f2 = new AffichageDiffusion((I_2D) unFragment, "Diffusion");
        f2.setLocation(200, 0);
        f2.show();

        JFrame f3 = new AffichageIdFragment((I_2D) unFragment,
                                            noFragmentInitial);
        f3.setLocation(400, 0);
        f3.show();

    } else {

    }

}

public void simuler(Fragment unFragment, long time) {

    // Pour eviter de le calculer NB_ITERATION fois...
    if (Parametres.CALCULER_OMEGA_OPTIMAL) {
        Parametres.calculerOmegaOptimal();
    }

    for (int i = 0; i < Parametres.NB_ITERATIONS; i++) {

        if (Parametres.DIFFUSER) {
            unFragment.diffuser();
        }

        unFragment.changerEtats();

        if (Parametres.CREER_GIF_NOIR_BLANC) {
            unFragment.creerGifNoirBlanc(i, time);

        }
        if (Parametres.CREER_GIF_COULEUR) {
```

```
                unFragment.creerGifCouleur(i, time);

            }

            // Inserer fonction pour calculer Complexite Morphologique
        }
    }


    public static void copy(File source, File dest) throws IOException {
        FileChannel in = null, out = null;
        try {
            in = new FileInputStream(source).getChannel();
            out = new FileOutputStream(dest).getChannel();

            long size = in.size();
            MappedByteBuffer buf = in.map(FileChannel.MapMode.READ_ONLY, 0,
                                          size);

            out.write(buf);

        } finally {
            if (in != null) {
                in.close();
            }
            if (out != null) {
                out.close();
            }
        }
    }


}
```

Bibliographie

(1) Coxe A.M and Clifford A.R. Fuzzy hexagonal automata and snowflakes. *Computers and Graphics*, 27(3) :447–454, 2003.

(2) G.R. Andrews. *Foundations of Multithreaded, Parallel, and Distributed Programming.* Addison Wesley, 2000.

(3) R. Antonin. Modélisation numérique des processus de dissolution dans les brèches hydrothermales : Applications aux brèches uranifères du bassin d'Athabasca, Province du Saskatchewan, Canada. Master's thesis, UQAM, 2004.

(4) auteur inconnu. Site internet de wikipedia, l'encyclopédie gratuite. `http://en.wikipedia.org/wiki/Spaceship_%28CA%29`, Mai 2005.

(5) P. Bak and K. Chen. A forest-fire model and some thoughts on turbulence. *Physics letters A*, 147 (5-6) :297–299, 1990.

(6) C. C. Barton and P.R. La Pointe. *Fractals in Petroleum Geology and Earth Processes.* Plenum, New York, 1995.

(7) R. Bédard. Notes de cours de MAT4112 : équations aux dérivées partielles. Mai 2003.

(8) J.S. Cameron. Site internet du college of wooster. Mars 2001.

(9) R.J. Charbeneau. Multicomponent exchange and subsurface solute transport : characteristics, coherence and the Riemann problem. *Water resources Res.*, 24 :57–64, 1988.

(10) B. Chopard, P. O. Luthi, and P.-A. Queloz. Cellular automata model of car traffic in two-dimensional street networks. *J. Phys. A*, 29 :2325–2336, 1996.

(11) P. Colella and P.R. Woodward. The piecewise parabolic method (ppm) for gas-dynamics simulations. *J. Comput. Phys.*, 54 :174, 1984.

(12) J. Crank. *Mathematics of diffusion, 2nd edition.* Oxford University Press, Oxford, 1975.

(13) P. Cubillas, M. Prieto, E.H. Oelkers, and S. Köhler. How do mineral coatings affect dissolution rates ? *Goldschmit Conference*, 68 :143, 2004.

(14) L. Empereur and T. Villemin. Obsifrac : database-supported software for 3d modeling of rock mass fragmentation. *Computers and Geosciences*, 29 :173–181, 2003.

(15) D. Eppstein. Searching for spaceships. *More Games of No Chance, MSRI publications*, 42 :433–453, 2000.

(16) M. Ferer, G.S. Bromhal, and D.H. Smith. Fractal dimension and avalanches invasion percolation : the effect of aspect ratio. *Physica A*, 334(1-2) :22–38, 2004.

(17) R.C. Fletcher and A.W. Hoffmann. Simple model of diffusion and combined diffusion-infiltration metasomatism. *Geochimical Transport and Kinetics*, 1 :243–259, 1974.

(18) U. Frish, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier-Stokes equation. *Phys. Rev. Lett.*, 56 :1505–1507, 1986.

(19) R. Furic. Communication privée. Juin 1995.

(20) M. Gardner. Mathematical games : The fantastic combinations of John Conway's new solitaire game life. *Scientific American*, pages 120–123, October 1970.

(21) S.K. Godunov. A difference scheme for numerical computation of discontinuous solutions of hydrodynamics equations. *Math. Sbornik.*, 47 :271–306, 1959.

(22) J.A. Grant. The isocon diagram – a simple solution to Gresens' equation for metasomatic alteration. *Economic Geology*, 81 :1976–1982, 1986.

(23) R.L. Gresens. Composition-volume relationships of metasomatism. *Chem. Geol.*, 2 :47–65, 1967.

(24) B. Guy. Mathematical revision of Korzhinskii's theory of infiltration metasomatic zoning. *Eur. J. Mineral.*, 5 :317–339, 1993.

(25) P.B. Hansen. *Studies in Computational Science : Parallel Programming Paradigms.* Prentice Hall, New Jersey, 1995.

(26) F. Higuera and J. Jimenez. Boltzmann approach to lattice gas simulations. *Europhys. Lett.*, 9(7) :663–668, 1989.

(27) P. Hébert and M. Panneton. Notes de cours de math-303 : dérivées partielles. 1998.

(28) J.-Y. Josnin, H. Jourde, P. Fénart, and P. Bidaux. A three dimensional model to simulate joint networks in fractured rocks. *Canadian J. of Earth Sci.*, 39 :1443–1455, 2002.

(29) M. Jébrak. Hydrothermal breccias in vein-type ore deposits : A review of mechanisms, morphology and size distribution. *Ore geology reviews*, 12 :111–134, 1997.

(30) M. Jébrak. Cours sur l'hydrothermalisme, 1998.

(31) M. Jébrak. Les brèches hydrothermales : Géométrie, physique et métallogénie. *Cours présenté au Congrès de l'Association des Géologues et des Géophysiciens du Québec*, Montréal, Canada, avril 1995.

(32) M. Jébrak and M. Lalonde. The shape of fragments in dissolution processes : fractal analysis of virtual breccias. *Journal of Structural Geology*, Soumis pour publication.

(33) D. Kandhai, D. Hlushkou, A.G. Hoekstra, M.A. Sloot, H. Van As, and U. Tallarek. Influence of stagnant zones on transient and asymptotic dispersion in macroscopically homogeneous porous media. *Phys. Rev. Lett.*, 88(23) :234501, 2002.

(34) D.S. Korzhinskii. *Theory of Metasomatic Zoning.* Clarendon Press., Oxford, 1970.

(35) A.C. Lasaga. *Kinetic theory in the earth sciences.* Princeton Univ. Press, Princeton, 1998.

(36) P.C. Lichtner. Continuum models for simultaneous chemical reactions and mass transport in hydrothermal systems. *Geochim. Cosmochim. Acta*, 49 :779–800, 1985.

(37) B. Mandelbrot. *The Fractal Geometry of Nature.* W.H. Freeman and Company, New York, 1983.

(38) N.S. Martys, J.G. Hagedorn, D. Goujon, and J.E. Devaney. Large scale simulations of single and multi-component flow in porous media. *SPIE : The International Society for Optical Engineering*, 3772 :205–213, Juillet 1999.

(39) J.E. Olson. Joint pattern development : effects of subcritical crack growth and mechanical crack interaction. *Geophys. Res*, 98, B7 :12251–12265, 1993.

(40) P. Ortoleva, E. Merino, C. Moore, and J. Chadam. Geochemical self-organization : Reaction-transport feedbacks and modelling approach. *Am. J. Sci.*, 287 :979–1007, 1987.

(41) N.H. Packard and S. Wolfram. Two-dimensional cellular automata. *Journal of Statistical Physics*, 38 :901–946, 1985.

(42) Y.H. Qian, D. d'Humières, and P. Lallemand. Lattice BGK for Navier-Stokes equation. *Europhysics Letters*, 17(6) :479–484, 1992.

(43) Office québécois de la langue française. Site du grand dictionnaire terminologique. http ://www.granddictionnaire.com/, 1987, 1995, 1997.

(44) A. Rosenfeld and J.L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM*, 13(4) :471–494, 1966.

(45) A. Roulcau. Un survol de modèles mathématiques basés sur les équations différentielles en géologie. Mars 2005.

(46) T. Rowland. Site internet de mathworld. Mai 2005.

(47) R. Rucker and J. Walker. Site internet du cellab. http ://www.fourmilab.ch/cellab/.

(48) M. Sahimi and T.T. Tsotsis. Dynamic scaling for the fragmentation of reactive porous media. *Phys. Rev. Lett*, 59 :888–891, 1987.

(49) M. Savard. Rapport de stage : Simulation numérique d'échanges fluides-roche. CNRS, Juillet 1983.

(50) P. Shut. Site internet de agriculture et agroalimentaire canada. http ://www.agr.gc.ca/, Mars 2001.

(51) W.G. Strang and G.J. Fix. *An Analysis of the Finite Element Method.* Prentice Hall, New Jersey, 1973.

(52) A. Takeda, S. Cox, and A.J. Payne. Parallel numerical modelling of the Antarctic Ice Sheet. *Computers and Geosciences*, 28 :723–734, 2001.

(53) D.L. Turcotte. Fractals in geology and geophysics. *Pure and Applied Geophysics*, 131 :171–196, 1989.

(54) S. Ulam. *Random Processes and Transformations.* MIT Press, Cambridge, 1950.

(55) D. Vanbergue and A. Drogoul. Approche multi-agent pour la simulation urbaine. In *Actes des 6e Journées CASSINI 2002*, 2002.

(56) H. von Koch. Influence of stagnant zones on transient and asymptotic dispersion in macroscopically homogeneous porous media. *Archiv för Matemat., Astron. och Fys.*, 1 :681–702, 1904.

(57) J. von Neumann. *Theory of Self-Reproducing Automata.* University of Illinois Press, Urbana, 1966.

(58) D. Wilkinson and J.F. Willemsen. Invasion percolation : A new form of percolation theory. *J. Phys. A : Math. Gen*, 16 :3365–3376, 1983.

(59) S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55 :601–644, 1983.

(60) S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10 :1–35, 1984.

(61) S. Wolfram. *A new kind of science.* Wolfram Media inc., Canada, 2002.