Modélisation multi-échelles de réservoir et calage d'historique de production

Caroline Gardet

Modélisation Multi-Echelles de Réservoir et Calage d'historique de production

Par Caroline Gardet

Thèse de doctorat de l'université Pierre et Marie Curie

Spécialité Geosciences

Dirigée par Mickaele Le Ravalec et Erwan Gloaguen

Présentée et soutenue publiquement le 14/11/2014

Devant un jury composé de :

Rapporteurs :	Chantal de Fouquet	
	Philippe Renard	
Examinateurs :	Sophie Violette	
	Guillaume Caumon	
	Mickaele Le Ravalec	*Directrice de thèse*
	Erwan Gloaguen	*Co-directeur*

ED 398 – Géosciences, Ressources Naturelles et Environnement
Université Pierre et Marie Curie
4 Place Jussieu, 75005 Paris

Remerciements

Tout d'abord je tiens à remercier Mickaele Le Ravalec, ma directrice de thèse, qui a cru en mes capacités et qui m'a guidée et soutenue pendant ces années de thèse. Merci de même à Erwan Gloaguen qui m'a accueillie à l'INRS pendant quelques semaines et qui m'a distillé de précieux conseils pendant ces 3 ans. Merci à toute l'équipe de réservoir.

Je suis également reconnaissante à Chantal de Fouquet, Maitre de recherche à l'Ecole des Mines de Paris, et à Philippe Renard, Professeur à l'Université de Neuchâtel en Suisse, pour m'avoir fait l'honneur d'être les rapporteurs de ce mémoire. Merci à Sophie Violette et Guillaume Caumon d'avoir accepté d'être membre de mon jury.

Ma deuxième salve de remerciements s'adresse plus particulièrement à mes coreligionnaires : Benjamin mon collègue de bureau qui a gardé mon cerveau au frais pendant 3 ans et avec qui j'ai partagé moult débats passionnants, Arthur pour avoir gardé une bonne humeur constante et alimenté nos débats, Noelia pour m'avoir fait découvrir le Pérou et la générosité sud-américaine, Ratiba, Katia, François, les anciens thésards pour leur accueil quand j'ai commencé au département réservoir, Sandra pour nos discussions sur le théâtre et la danse. Une mention spéciale à Claire, Antoine et Mathieu qui se reconnaitront. J'en oublie plein d'autres évidemment.

Le meilleur pour la fin, merci à ma famille qui m'a soutenu tout au long de mes très longues études. A mes parents qui m'ont toujours poussée plus haut et à mon petit frère qui a même relu ce mémoire !

Résumé

La modélisation de réservoir est un outil clé pour évaluer le potentiel des gisements d'hydrocarbures, et optimiser leur exploitation. A partir des modèles, on va prédire le comportement du réservoir, ce qu'il peut produire et sous quelles conditions. Or les réservoirs sont issus de multiple processus géologiques complexes. Ils présentent des hétérogénéités de taille et de forme très variées, ce qui les rend difficiles à évaluer, d'autant qu'on ne dispose pas d'une vue directe sur le sous-sol. Il faut construire ces modèles à partir des données disponibles, peu nombreuses et d'origines très différentes.

Dans ces travaux de thèse, nous proposons deux modèles multi-échelles de réservoir : l'un issu de la géostatistique à deux points, l'autre issu de la synthèse de texture et des algorithmes multipoints. Ces algorithmes sont détaillés sur deux échelles, mais le passage à plus de deux échelles est direct.

Le premier algorithme est une version multi-échelles de la simulation séquentielle Gaussienne, basé donc sur l'utilisation d'un variogramme pour décrire les variations spatiales des propriétés pétrophysiques qu'elles soient discrètes ou continues. Dans cette technique, on simule une variable à l'échelle fine, et sa moyenne à une échelle plus grossière. Le premier point sur lequel nous nous sommes penchés est le calcul de la covariance de la variable moyennée et de la covariance croisée à partir de la variable à l'échelle fine. Ensuite, nous avons proposé comme schéma, premièrement de simuler la variable moyennée à l'échelle grossière, puis deuxièmement de simuler la variable à l'échelle fine en prenant comme contrainte la réalisation à l'échelle grossière. Après avoir montré comment on combinait l'algorithme avec la méthode de la déformation graduelle, on a illustré le potentiel de la méthode sur un cas de calage d'historique synthétique. Cependant, comme toutes les méthodes qui utilisent un variogramme, notre méthode rencontre quelques difficultés pour simuler des réservoirs complexes comme ceux contenant des chenaux. C'est pourquoi nous avons cherché à appliquer le concept multi-échelles à une catégorie assez nouvelle d'algorithmes, les méthodes statistiques multipoints.

Le deuxième algorithme est une adaptation d'un algorithme de synthèse de texture aux problèmes de réservoir. C'est un algorithme de simulation multipoints qui nécessite donc l'utilisation d'une image d'entrainement. Nous présentons donc les étapes successives de l'algorithme, puis l'analyse de sensibilité des paramètres, et surtout comment on intègre les données dures. Nous terminons par deux techniques de diminution du temps de calcul, point très problématique de ce genre d'algorithme. Pour illustrer les performances de cet algorithme, nous le testons sur des images variées que l'on peut espérer trouver dans un réservoir, des chenaux, des méandres, des grains et des fractures. Les résultats sont assez satisfaisants, mais deux points difficiles sont relevés : le temps de calcul et la reproduction de très grands objets.

Dans le troisième algorithme, on passe à la synthèse de texture multi-échelles justement pour pallier les deux problèmes soulevés précédemment. La méthode se déroule en deux étapes de simulation. La première simulation à l'échelle grossière, permet 1) de capturer les grandes hétérogénéités avec un petit template 2) d'être très rapide car les grilles considérées sont de basse résolution. La deuxième simulation à l'échelle fine permet d'ajouter des détails sur les grandes structures préalablement simulées à l'échelle grossière. Après une brève comparaison avec l'algorithme mono-échelle, on teste les deux techniques d'accélération du temps de simulation utilisées sur l'algorithme précédent.

Table des matières

Table des Figures

Introduction Générale

1.1. Qu'est-ce qu'un réservoir ?

Une roche-réservoir est une roche poreuse et perméable dans laquelle s'accumulent des hydrocarbures. En effet sous l'effet de la pression, les hydrocarbures migrent de leur roche de formation (dite roche-mère) vers la surface à travers diverses strates de roches. Ils peuvent se retrouver piégés lorsqu'ils rencontrent une strate imperméable, de type argileuse, ou un dôme de sel. Ce type de gisement, composé d'une strate imperméable au-dessus d'une strate poreuse et perméable saturée en hydrocarbures, constitue ce que l'on appelle les gisements conventionnels, illustrés Figure 1.

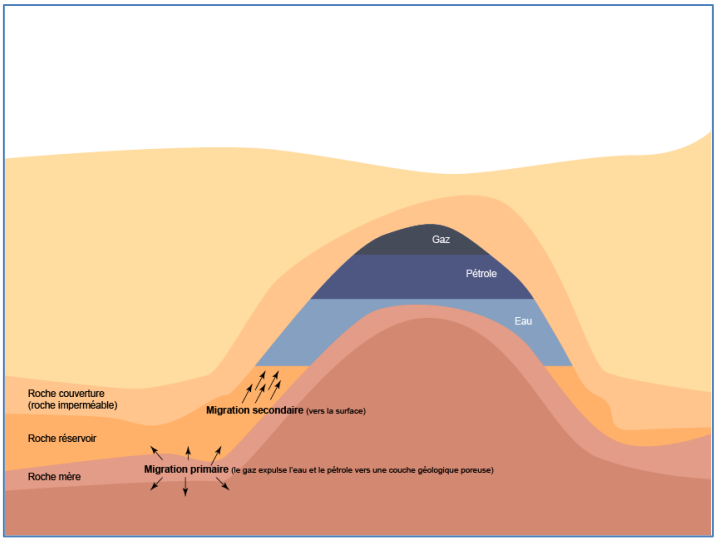


Figure 1 Schéma d'un gisement conventionnel d'hydrocarbure. Source : **http://www.connaissancesdesenergies.org**

On appelle gisement non-conventionnel un gisement où par exemple la roche-mère est aussi la roche-réservoir, comme c'est le cas pour le gaz et l'huile de schiste. Les schistes bitumineux font aussi partie de ces gisements non-conventionnels. Ce sont des roches sédimentaires de très fine granulométrie contenant des hydrocarbures immatures. C'est-à-dire que la roche-mère n'a pas été soumise aux conditions nécessaires à la transformation de la matière organique, ou pas assez longtemps. On peut aussi trouver les réservoirs d'huiles lourdes, le gaz de réservoir compact et le gaz de charbon. Tous ces gisements non-conventionnels sont schématisés très simplement dans la Figure 2.

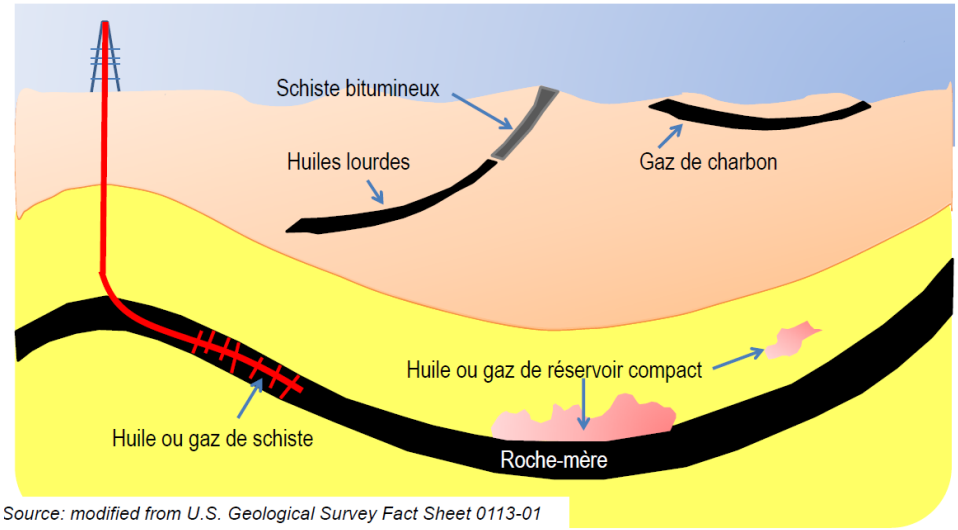


Figure 2 Schéma de gisements non-conventionnels

On voit donc que les réservoirs sont situés dans des couches géologiques issues de processus complexes et donc les réservoirs sont des objets complexes. La formation de ces strates fait intervenir des phénomènes de déposition, d’érosion, de diagenèse, de fracturation, etc. Il en résulte des hétérogénéités de tailles et de formes multiples. Par exemple, on peut avoir des chenaux, des lobes ou encore des méandres et à l'intérieur, des phénomènes de sédimentation et d'accrétion. L’idée derrière cette thèse est d’utiliser l’architecture multi-échelles du réservoir pour construire un modèle sur différent niveaux de résolution, modifiable à toutes les échelles indépendamment les unes des autres afin d’apporter une image plus représentative de ces objets que l’on ne peut voir.

1.2. Qu’est-ce qu’un modèle de réservoir ?

Le but de la modélisation de réservoir est de fournir une représentation numérique du réservoir, la plus exacte/représentative possible d’un point de vue géologique. On cherche à savoir dans le réservoir, la saturation en hydrocarbures (la fraction du vide occupée par le pétrole), la porosité (l’espace libre occupé par un fluide dans une roche), la perméabilité (la capacité d’une roche à laisser s’écouler un fluide), le volume d’argile,… Ces propriétés vont servir à estimer le volume d’hydrocarbures disponible et récupérable. Le réservoir est donc découpé en mailles (cube, tétraèdre, …), et dans chaque maille on veut connaitre la valeur de ces propriétés pétrophysiques. Un exemple de réservoir découpé en grille est illustré Figure 3. Cette grille numérique est ensuite envoyée en entrée du simulateur d’écoulement pour prédire les écoulements de fluides dans le réservoir selon différents scénarios d’exploitation. Généralement la propriété pétrophysique la plus importante est la perméabilité, sa distribution spatiale conditionne le chemin des écoulements et ses principaux obstacles. De nombreux travaux traitent des hétérogénéités dans les modèles de réservoir pour comprendre l’impact des architectures complexes sur les écoulements de fluides (Hewett, 1986; Koltermann and Gorelick, 1996; De Marsily *et al.*, 2005).

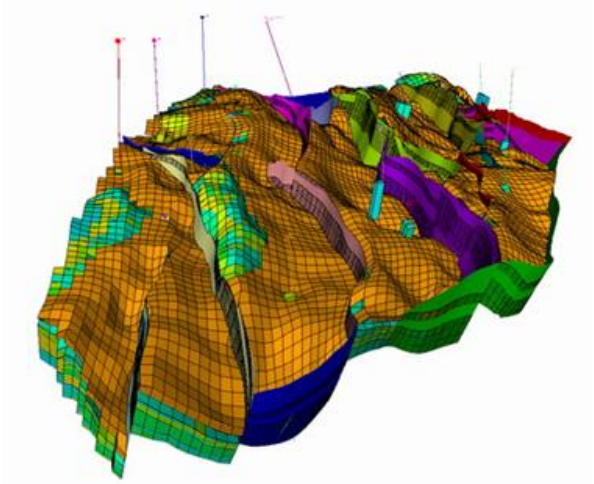


Figure 3 Exemple de modèle de réservoir. Source : http://hw.tpu.ru/en/

1.3. Quelles sont les données utilisées ?

Une des difficultés majeures rencontrées en modélisation de réservoir est la diversité et la rareté des données disponibles. On peut distinguer 4 types de données, 3 sont dits statiques car ces données ne varient pas dans le temps, et un dernier type est dit dynamique car il varie au cours de l'exploitation du champ pétrolier. Dans les données statiques, il y a premièrement, les données sismiques qui couvrent le volume étudié mais qui ont une résolution assez faible (de 5 à 10 m lors des campagnes sismiques les plus précises). Elles sont obtenues en mesurant les échos issus de la propagation d'une onde sismique provoquée en surface. Ces échos sont dus aux changements de propriétés des roches entre deux strates, ces interfaces deviennent des réflecteurs pour les ondes. Un exemple de sismique 2D est donné Figure 4.

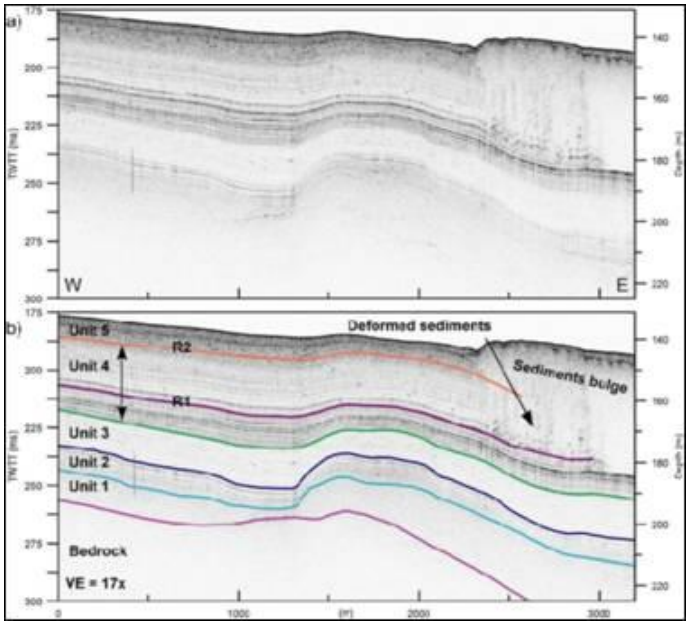


Figure 4 Données sismiques. Sources : http://www.u-picardie.fr/beauchamp/GazSchiste

On voit que la sismique permet de définir les différentes strates ou encore la structure du sous-sol, et d'apporter des informations sur la nature des strates. La sismique définit l'enveloppe géométrique du réservoir.

Deuxièmement on peut citer les données de puits (carottes, diagraphies, cuttings, etc.), qui sont plus précises, leur résolution est centimétrique pour les carottes à décimétrique pour les diagraphies. Cependant elles ne sont disponibles qu'au niveau des puits et sont donc éparses et seulement valables autour des puits. Un exemple de diagraphie est montré Figure 5. Une diagraphie consiste à mesurer le long du puits (pendant ou après un forage) certaines propriétés des roches traversées. La diagraphie représente donc un ensemble de caractéristiques du sous-sol en fonction de la profondeur. On peut mesurer la teneur en hydrocarbure, la porosité, la perméabilité, la vitesse de propagation d'une onde, la résistivité, etc.

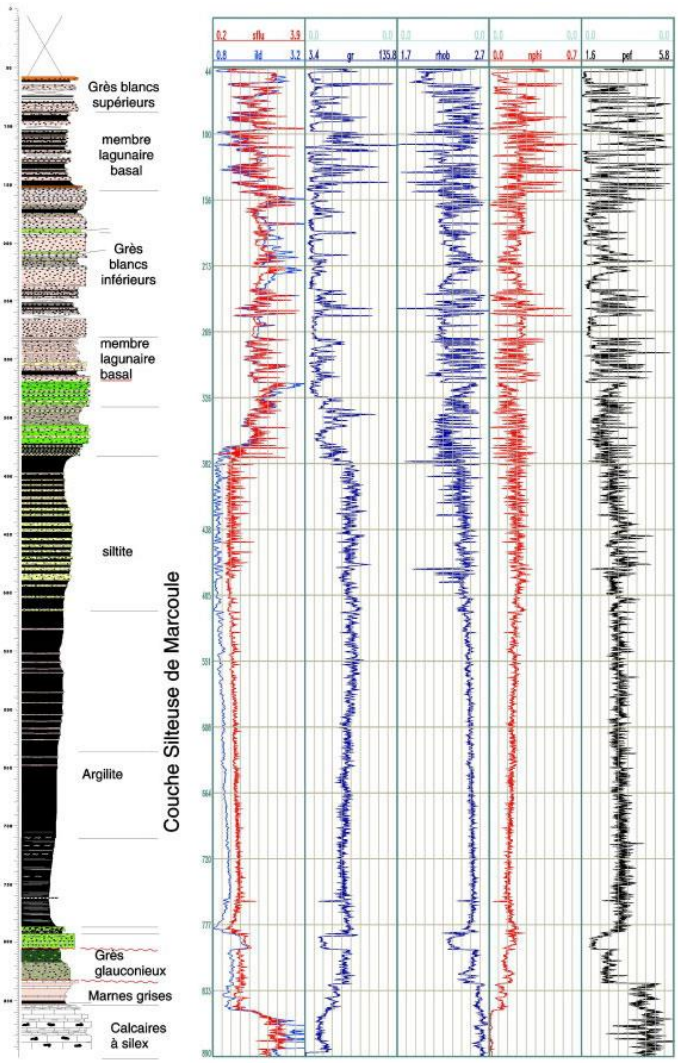


Figure 5 Exemple de diagraphie. Source : http://www.geologues-prospecteurs.fr

Enfin citons comme données statiques disponibles, les informations qualitatives déduites de l'analyse du contexte régional d'un réservoir, l'étude d'analogues et l'utilisation de concepts géologiques tels que la stratigraphie séquentielle ou bien, en quelque sorte, les images d'entraînement dans les statistiques multipoints.

Ces données statiques sont intégrées dans le réservoir par des techniques géostatistiques. Les méthodes statistiques à deux points sont généralement utilisées pour simuler les réalisations nécessaires. Elles impliquent la définition d'un variogramme qui mesure la dissemblance entre deux points dans l'espace (un état de l'art plus détaillé se trouve en 3.1). Cependant comme le variogramme est restreint à l'analyse des statistiques entre 2 points seulement, il ne permet pas de caractériser des hétérogénéités géologiques complexes comme les chenaux, les méandres et

les lobes. La nécessité de mieux reproduire la géométrie de ce type d'hétérogénéité a mené au développement de nouveaux types de méthodes (Strebelle, 2002), dont les méthodes de simulation multipoints de type « object-based » et « pixel-based » (de même un état de l'art plus détaillé sur les méthodes de simulation multipoints est disponible dans la section 4.1).

Terminons par les données dynamiques, que sont les débits d'eau et d'hydrocarbures aux puits, les pressions aux puits en tête ou en pied, les temps de percée (water-cut), etc. Ces mesures varient au cours du temps, car lors de l'extraction du pétrole la pression diminue dans le réservoir. On a alors recours à de l'injection de vapeur, d'eau ou autres composés chimiques pour maintenir la pression dans le réservoir et faciliter la récupération. Ces données apportent énormément d'informations sur le réservoir mais ce sont des données indirectes, elles ne sont pas directement liées aux propriétés mesurables de la roche comme la porosité ou la perméabilité. Leur intégration dans un modèle de réservoir est un problème inverse, généralement résolu par optimisation (Yeh, 1986 ; Sun, 1995). En d'autres termes on doit passer par une phase de simulation d'écoulement des fluides sur le modèle de réservoir, pour obtenir une courbe de production que l'on va pouvoir comparer aux données réelles de production. Les problèmes directs et inverses sont détaillés dans la première section de ce rapport (2.1 et 2.2).

1.4. Intégration des données

L'importance de chaque information n'est pas dans son utilisation seule, mais dans ce qu'elle apporte dans l'analyse de l'ensemble des données. La caractérisation du réservoir doit se faire en utilisant toutes les données disponibles. Rappelons que les modèles de réservoirs sont des grilles assez finement découpées : elles comprennent quelques millions de mailles. Le but de la caractérisation est de déterminer les valeurs de perméabilité et porosité dans chaque maille. Toutes les valeurs non connues de ces propriétés constituent les inconnues du problème. Il y a donc beaucoup plus d'inconnues (le même nombre que de mailles au moins) que de données, ce qui rend le problème inverse indéterminé, et aussi le plus souvent, mal-posé. De fait, il peut ne pas y avoir de solution, il peut y avoir plusieurs solutions, ou encore, même s'il y a une solution, cette solution peut être très sensible aux variations des données. Par ailleurs, le processus d'optimisation est très coûteux en temps de calcul, car il demande un grand nombre d'itérations et pour chaque itération une simulation d'écoulement. Une seule simulation d'écoulement peut requérir quelques heures de calculs. Une des possibilités utilisées pour éviter ces inconvénients est de réduire le nombre de paramètres par une technique de paramétrisation adéquate.

L'"upscaling" est une des solutions existantes. Le modèle fin de réservoir est converti en un modèle plus grossier contenant moins de mailles. L'optimisation peut alors être effectuée sur le modèle grossier au lieu du modèle fin, ce qui permet de réduire le nombre de paramètres inconnus. L'inconvénient de cette approche est qu'elle ne permet pas de conserver le lien entre les échelles, i.e., la continuité de l'échelle fine à l'échelle grossière. Le modèle grossier optimisé respectera peut-être les données dynamiques, mais pas les données statiques, et le modèle fin respectera les données statiques mais pas les données dynamiques. D'autres techniques ont été proposées pour réduire le nombre de paramètres. Par exemple, les techniques basées sur les transformée de Karhunen-Loève (Efendiev *et al.*, 2006 ; Romary, 2010), ou sur les transformées en cosinus discrètes (Jafarpour and MacLaughlin, 2007), ou sur les ondelettes (Sahni and Horne, 2005). Ces méthodes permettent de réduire les dimensions de la base dans laquelle est décrit le réservoir. Dans ces cas-là, l'optimisation est réduite à l'ajustement des paramètres dans la nouvelle base. Cependant, il reste le problème de la variabilité spatiale. En effet, ces techniques ne permettent pas d'assurer le respect de la variabilité spatiale déduite des données

statiques. C'est cet inconvénient qui a motivé le développement de techniques de paramétrisation basées sur la géostatistique. Leur principal avantage est de diminuer le nombre de paramètres tout en préservant la variabilité spatiale. Parmi toutes ces techniques, on peut distinguer différentes méthodes : la méthode des points pilotes (Marsily *et al.*, 1984 ; RamaRao *et al.*, 1995), la calibration séquentielle (Gomez-Hernandez *et al.*, 1997), la méthode de déformation graduelle (Hu, 2000), la méthode des perturbations de probabilités (Caers, 2003) ou encore la méthode de perturbation par cosimulation (Le Ravalec-Dupin et Da Veiga, 2011). D'autres méthodes géostatistiques de paramétrisation considèrent une déformation à un niveau plus grossier. Elles ajustent localement la moyenne de la perméabilité au lieu des fluctuations de la perméabilité autour de la moyenne (Fenwick *et al.*, 2005 ; Le Ravalec-Dupin, 2010 ; Gervais and Roggero, 2010).

Ces problématiques à propos des niveaux de résolution conduisent à se pencher sur la paramétrisation multi-échelles. Ces techniques semblent bien adaptées pour capturer correctement le caractère multi-échelles intrinsèque à la fois du réservoir et des données. Elles offrent la possibilité de modifier un modèle de réservoir à différentes échelles et de prendre en compte les données à leur juste niveau de résolution. Tran *et al.* (1999) ont proposé le schéma suivant. Premièrement, le modèle fin de réservoir est upscalé à l'échelle grossière. Puis, le processus d'optimisation permet de déterminer le modèle grossier optimal. Enfin, ce modèle grossier optimal est ramené à l'échelle fine (processus dit de « downscaling ») par simulation séquentielle Gaussienne (Goovaerts, 1997). Cependant, le modèle fin de réservoir ainsi obtenu peut ne pas respecter les données dynamiques. Aanonsen et Eydinov (2006) ont complété ce schéma avec un second processus d'optimisation à l'échelle fine afin d'éviter ce piège.
Dans ce mémoire, nous revenons sur l'idée introduite par Tran *et al.* (1999) en développant une technique d'optimisation combinée avec une paramétrisation multi-échelles capable de gérer à la fois les propriétés pétrophysiques continues (perméabilité, porosité, *etc*.) et discrètes (faciès, *etc*.).

Tous les chapitres présentant les modèles développés commencent par une bibliographie.
Ce manuscrit s'organise comme suit.

Chapitre 2 : Problèmes direct et inverse - Déformation graduelle

Dans ce chapitre nous détaillons ce qui sera commun aux deux approches développées. Le problème d'écoulement direct est l'ensemble des équations qui régissent les écoulements en milieu poreux. Le problème inverse ou calage d'historique de production, est la méthode utilisée pour intégrer des données qui ne sont pas directement reliées à la propriété modélisée. Ces problèmes sont indépendants de la méthode de modélisation de réservoir choisie tant qu'elle est compatible avec une méthode de perturbation pour l'inclure dans le calage.
La méthode de déformation graduelle, choisie pour tous les calages, est détaillée en fin de chapitre.

Chapitre 3 : Simulation séquentielle Gaussienne

Dans ce chapitre nous présentons le modèle issu de la statistique à deux points. Dans un premier temps on rappelle ce qu'est l'algorithme de simulation séquentielle Gaussienne, puis on explique l'adaptation multi-échelles que l'on a développée. On montre un exemple pour une propriété continue, puis on regarde comment on passe aux propriétés discrètes. En fin de chapitre, on introduit une application numérique, qui montre le potentiel de la méthode proposée.
Cette partie des travaux de thèse à fait l'objet d'une publication :

Gardet, C., Le Ravalec, M., Gloaguen, E., 2014, Multiscale Parameterization of Petrophysical Properties for Efficient History-Matching, Mathematical Geosciences, 46(3), 315-336, doi: 10.1007/s11004-013-9480-3

Chapitre 4 : Synthèse de texture Mono-Echelle

Dans ce chapitre, on présente l'adaptation de l'algorithme de synthèse de texture dans le contexte de modélisation de réservoir et notamment sur l'intégration des données dures. On détaille l'algorithme pas à pas, avant de faire une analyse de sensibilité des paramètres. On s'intéressera aussi au calcul de la mesure de distance entre deux motifs. Ce chapitre se termine par le test de deux techniques pour diminuer les temps de calcul, et quelques résultats sur des images de propriétés continues, présentant une architecture complexe.
Ces travaux ont donné lieu à la rédaction d'un article :

Gardet, C., Le Ravalec, M., Gloaguen, E., (soumis), An hybrid algorithm combining multiple-point simulation and texture synthesis for generating conditional reservoir models, Stochastic Environmental Research and Risk Assessment.

Chapitre 5 : Algorithme Multipoint Multi-Echelles

Dans ce chapitre on détaille l'approche multipoints multi-échelles et on montre qu'elle résout les problèmes soulevés au chapitre 4. On explique l'obtention de l'image grossière de l'algorithme multi-échelle pour des propriétés continues et discrète. On termine de même avec le test des deux techniques d'accélération de l'algorithme.
Ces travaux ont aussi donné lieu à la rédaction d'un article :

Gardet, C., Bouquet, S., Le Ravalec, M., (soumis), Multiscale simulation of geological formations with multipoint statistics, Stochastic Environmental Research and Risk Assessment.

Conclusion Générale - Perspectives

Cette partie nous permet de prendre du recul sur la thèse et de résumer les approches développées, de tirer les conclusions sur les résultats obtenus et de proposer quelques futurs axes de recherche.

Chapitre 2

2. Problèmes direct et inverse - Déformation graduelle

Cette partie présente les problèmes direct et inverse ainsi que la méthode de déformation graduelle. Le problème direct est défini à partir des équations décrivant les écoulements de fluides en milieu poreux. Le problème inverse explique comment à partir de données indirectes, on peut caractériser la perméabilité au sein du réservoir. On définit ce qu'est une fonction objectif et on explique le schéma de la boucle de calage mis en place pour le calage d'historique. Enfin, on rappelle les principes de la méthode de déformation graduelle : il s'agit d'une méthode de paramétrisation qui permet de modifier un modèle de réservoir à partir d'un nombre réduit de paramètres.

2.1. Problème direct

Les écoulements dans le réservoir sont décrits par un ensemble d'équations. Les lois utilisées sont principalement la loi de conservation de la masse et les lois de Darcy. Pour simplifier les calculs, on considère l'écoulement de deux phases incompressibles et immiscibles dans un milieu dont la perméabilité est isotrope.
Les lois de Darcy s'écrivent alors :

$$u_o = -\frac{k_{ro}(s_o)}{\mu_o} K\nabla(P_o)$$
$$u_w = -\frac{k_{rw}(s_w)}{\mu_w} K\nabla(P_w)$$

Equation 1

Les indices *w* et *o* se rapportent à l'huile et à l'eau, respectivement. Ce sont les deux seules phases présentes dans le réservoir. *P* fait référence à la pression et *s* à la saturation. μ représente la viscosité et *u* la vitesse d'écoulement de la phase donnée. *K* est le tenseur de perméabilité symétrique du milieu considéré. La perméabilité relative k_r est une fonction de la saturation : elle est souvent donnée par une loi puissance, de type Corey (Corey, 1977) :

$$k_{rw}(s_w) = k_{rw\max}\left(\frac{s_w - s_{wir}}{1 - s_{wir} - s_{or}}\right)^{n_w}$$
$$k_{ro}(s_o) = k_{ro\max}\left(\frac{s_o - s_{or}}{1 - s_{wir} - s_{or}}\right)^{n_o}$$

Equation 2

k_{rwmax} est la perméabilité relative maximale pour l'eau, s_{wir} est la saturation irréductible à l'eau et s_{or} la saturation irréductible à l'huile. n_w et n_o sont les exposants de Corey pour l'eau et l'huile.

La fermeture du système est obtenue en supposant que le support poreux est complètement saturé et que la pression capillaire est donnée par :

$$\begin{aligned} s_o + s_w &= 1 \\ p_o - p_w &= p_c(s_w) \end{aligned}$$ **Equation 3**

La loi de conservation de la masse pour une phase i est donnée par :

$$\frac{\partial}{\partial t}(\rho_i s_i) + div\left(\sum_i \rho_i v_i\right) + \delta q_i = 0$$ **Equation 4**

Avec : ρ_i la densité de la phase i (eau ou huile), s_i la saturation de la phase i, δq_i le débit d'injection ou de production de la phase i, $\boldsymbol{v_i}$ la vitesse d'écoulement de la phase i.

On a alors un système d'équations couplées non-linéaires. Les conditions initiales dépendent de l'étude menée suivant que l'on est en exploration ou en production. En exploration on considérera le réservoir à l'équilibre de pression, alors qu'en production on connait les pressions au moment du commencement de l'exploitation. Les conditions aux limites sont généralement que le réservoir est imperméable à l'écoulement.

A partir de ces lois, pour un modèle de réservoir donné, peuplé par des propriétés pétrophysiques, on obtient des réponses dynamiques comme les pressions et les débits aux puits ou encore les saturations dans tout le réservoir à un temps donné.

2.2. Problème inverse – Calage d'historique de production

Le problème inverse consistant à intégrer les données de production tels les débits et les pressions est appelé, en ingénierie de réservoir, calage d'historique de production. Il consiste à identifier un modèle de réservoir, qui une fois donné en entrée au système d'équations d'écoulement, donne des réponses proches des données dynamiques connues. On distingue deux types de méthodes pour résoudre ce type de problème. Le premier regroupe les méthodes de type Monte Carlo ou probabilistes (Evensen, 1994, 2009). Ces méthodes sont fondées sur la génération d'un ensemble de réalisations de la variable recherchée, ce qui donne une distribution a priori. On résout le problème direct pour chaque réalisation jusqu'au temps où on a la première mesure. Chaque variable est alors mise à jour à partir de la théorie des filtres de Kalman. Ce processus est répété jusqu'à ce que toutes les données soient assimilées. Les filtres de Kalman ont beaucoup été utilisés. Cependant, il reste un certain nombre de questions à propos du sur-paramétrage, de la mauvaise quantification des incertitudes ou encore de la gestion des solutions non-physiques (Gu et Oliver, 2005). Le second type de méthodes regroupe les approches variationnelles (Jacquard, 1965). Dans ce cadre, la solution recherchée minimise une fonction que l'on appelle fonction objectif. Cette fonction quantifie l'écart entre les données dynamiques et les réponses simulées pour le modèle de réservoir. Ce processus revient à implémenter un algorithme d'optimisation itératif. A chaque itération, les paramètres du réservoir sont ajustés et on effectue une simulation d'écoulement. Ces paramètres sont modifiés jusqu'à ce que les réponses simulées correspondent aux données dynamiques.

Le schéma de calage utilisé à IFPEN (Figure 6) est décrit ci-dessous.

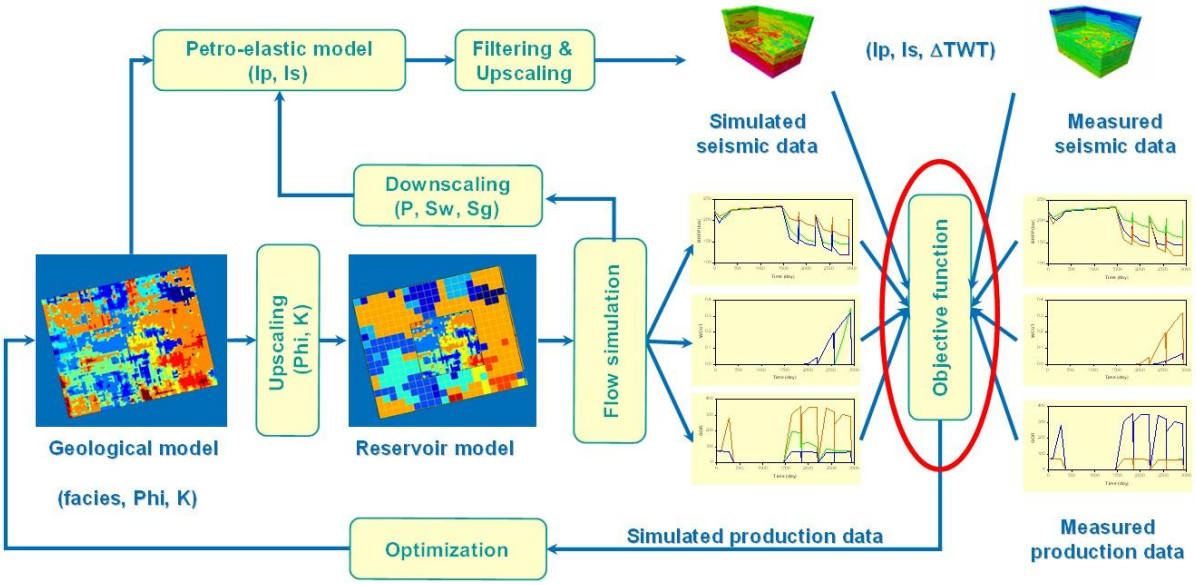


Figure 6. Schéma de calage d'historique de production.

Le but est de construire un modèle de réservoir qui respecte au mieux les données disponibles, aussi bien statiques (mesures aux puits,...) que dynamiques (débits, pression,...). La boucle de calage montre comment on contraint le modèle de réservoir. La description détaillée de cette boucle de calage est disponible dans Le Ravalec *et al.* (2012 OGST), ici nous ne faisons qu'un résumé.

- Premièrement, on construit un modèle géologique, sur une grille fine, à partir des données statiques telles que les mesures sur carotte, les diagraphies ou les modèles de sédimentation ou de déposition.
- La deuxième étape consiste à amener ce modèle vers une échelle grossière. On parle ici d' « upscaling ». L'intérêt est de réduire le temps de calcul lié à la simulation d'écoulement. On travaille donc sur une grille de moindre résolution où les propriétés pétrophysiques sont moyennées (à l'aide d'une moyenne arithmétique, géométrique,...).
- Dans un troisième temps, on donne ce modèle grossier en entrée au simulateur d'écoulement qui fournit alors des réponses en production (débits au cours de la production, pressions, "water-cut",...). A partir du modèle géologique et des résultats en pression et saturation issus du simulateur d'écoulement, on construit un modèle petroélastique, qui nous permet de simuler les attributs sismiques (données déduites de la sismique).
- La dernière étape de la boucle est le calcul de la fonction objectif, qui compare les données de production et les données sismiques simulées avec les données réelles.
- Pour réduire cette fonction objectif on perturbe le modèle géologique construit à l'étape 1 à l'aide de techniques telles que la méthode de déformation graduelle. On répète cette chaine de simulation jusqu'à ce que la fonction objectif passe sous un seuil préalablement défini.

La fonction objectif est définie par :

$$J(v) = \frac{1}{2}\left(g(v) - d_{obs}\right)^t \left(C_D\right)^{-1}\left(g(v) - d_{obs}\right) \qquad \textbf{Equation 5}$$

Elle mesure l'écart entre les réponses simulées $g(v)$ et les données dynamiques d_{obs}. g est l'opérateur qui permet de passer de l'espace des paramètres v aux réponses dynamiques : il s'agit ici du simulateur d'écoulement. $\mathbf{C}_D$ est la matrice de covariance qui quantifie les incertitudes expérimentales et théoriques. Généralement, un modèle de réservoir est composé de millions de mailles. Donc, il y a au moins autant d'inconnues que de mailles : chaque maille doit contenir une valeur de porosité, de perméabilité, de saturation, *etc*. D'un autre côté, le nombre de données dynamiques est nettement moins grand, ce qui rend le problème mal posé. Pour que le problème soit "mieux" posé, on utilise différentes techniques de régularisation. On peut par exemple ajouter de l'information dans la fonction objectif :

$$J(v) = \frac{1}{2}\left(g(v) - d_{obs}\right)^t \left(C_D\right)^{-1}\left(g(v) - d_{obs}\right) + \frac{1}{2}\left(v - v_0\right)^t \left(C_V\right)^{-1}\left(v - v_o\right) \qquad \textbf{Equation 6}$$

Le terme ajouté sur la droite évalue l'écart entre le modèle courant v et le modèle a priori v_o. La matrice de covariance $\mathbf{C}_V$ caractérise les incertitudes sur v_o.

Une autre technique de régularisation consiste à limiter l'espace de recherche. On peut alors utiliser des méthodes de paramétrisation géostatistiques telle que la méthode de déformation graduelle (Hu, 2000). Cette méthode, intégrée dans un processus d'optimisation, permet d'ajuster le paramètre de déformation de manière à minimiser la fonction objectif (cette méthode est détaillée section 2.3). Comme la méthode de déformation graduelle tient compte de l'information a priori, la fonction objectif est réduite au terme de différence entre les données. La méthode de déformation graduelle est rappelée plus loin.

De manière plus générale le problème d'optimisation s'écrit :

$$\min_{v \in \Omega} J(v) \qquad \textbf{Equation 7}$$

On rappelle que J est la fonction objectif et $\boldsymbol{v}$ les paramètres inconnus du modèles tels que la porosité, la perméabilité, *etc*. J est généralement fortement non linéaire dans le cadre du calage d'historique. Différentes méthodes d'optimisation sont utilisées pour résoudre ce problème : méthodes stochastiques (Gao *et al.*, 2007) ou déterministes (de Montleau *et al.*, 2006), méthodes globales (Bouzarkouna, 2012) ou locales (Bissell *et al.*, 1994), *etc*.
Les méthodes globales permettent un meilleur calage mais sont généralement trop couteuse en temps de calcul. Elles demandent un grand nombre d'évaluations de la fonction objectif et donc de simulations d'écoulement. Les méthodes locales sont souvent préférées. Nous détaillerons ici les 3 catégories en les expliquant rapidement.

- *Les méthodes de descente.*

  Ces méthodes s'appuient sur le calcul l'estimation du gradient ou de la Hessienne de J. En effet, la première étape de ces algorithmes est de partir d'un point initial et de le mettre à jour en se déplaçant dans la direction de descente. C'est cette direction de descente qui est calculée à partir des dérivées au premier et second ordre. Cependant dans les cas pétroliers ces dérivées n'ont jamais d'expression analytique, il faut donc calculer des approximations. On peut estimer le gradient par état adjoint. Cela consiste à créer un problème adjoint à la simulation d'écoulement et à résoudre les deux problèmes. Cependant peu de simulateur

d'écoulement proposent un code adjoint. On peut estimer le gradient par différences finies, en perturbant chaque paramètres. Mais cela implique de faire une évaluation de la fonction objectif pour chaque perturbation. Enfin citons la méthode bien connue de Quasi-Newton, qui s'appuie sur l'estimation de la Hessienne pour la direction de descente. De même la Hessienne est approximée à partir des calculs du gradient à chaque itérations.

- *Les méthodes de recherche par région de confiance*

Ces méthodes sont basées sur la construction de modèles approchés de J dans une région de confiance, recalculée à chaque itération. Ces modèles, souvent quadratiques, sont très facile à optimiser. Ils peuvent être calculés à partir de l'estimation du gradient et de la Hessienne comme dans SQP (Sequential Quadratic Programing) ou par interpolation (Bissell, *et al.*, 1994).

- *Les méthodes de recherche directe*

Ces méthodes ne nécessitent pas la connaissance des dérivées et sont très simples à implémentées. La première étape consiste à échantillonner la fonction objectif sur un certain nombre de points, puis de décider des prochaines évaluations à faire en se basant sur l'échantillonnage précédent. On retrouve deux types de méthodes : les méthodes par recherche directionnelle (Conn *et al.*, 2009) et les méthodes par simplexe (Nelder et Mead, 1965).

Une des principales préoccupations lors d'un calage d'historique de production est le temps de calcul, qui augmente avec le nombre d'itérations. Beaucoup de techniques ont été étudiées pour accélérer les techniques d'optimisation. L'approche développée ici se concentre sur la réduction des dimensions de l'espace de recherche, c'est-à-dire du nombre de paramètres inconnus. Comme expliqué précédemment, elle fait intervenir une paramétrisation multi-échelles des réalisations représentant les variations des propriétés pétrophysiques dans le réservoir. L'idée principale est de se donner la possibilité de changer les propriétés pétrophysiques à différentes échelles tout en préservant les liens entre les échelles. Pour résumer, on veut pouvoir changer les tendances d'une propriété à l'échelle grossière ou les variations plus fines de cette même propriété à une échelle plus fine. L'intérêt de cette approche est de gérer un nombre de paramètres qui dépend de l'échelle considérée. Ainsi, à l'échelle grossière, le nombre de paramètres inconnus est plus petit qu'à l'échelle fine. On cherche donc à déterminer d'abord les paramètres à l'échelle grossière. Puis, dans un deuxième temps, on envisage de raffiner le modèle de la propriété recherchée en variant les paramètres à l'échelle fine. Chaque fois qu'un modèle est raffiné, le nombre de paramètres augmente.

Dans le cas multi-échelles le modèle géologique se compose de plusieurs échelles de résolution différente qui peuvent être perturbées chacune indépendamment les unes des autres. Le schéma de calage utilisé est alors un peu différent de celui en Figure 6, il intègre ces nouvelles échelles. La Figure 7 illustre ce nouveau schéma.

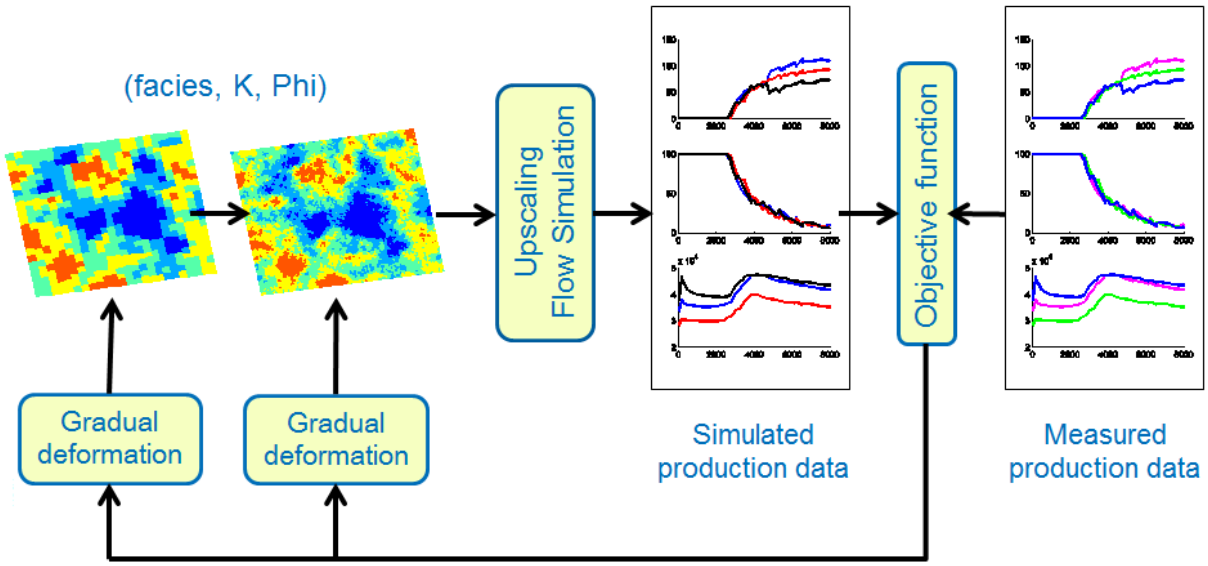


Figure 7 Schéma multi-échelles de calage d'historique de production.

La différence principale se situe dans la simulation du modèle géologique. Il est constitué de deux grilles de résolution différente intimement liées l'une à l'autre. Le calcul du modèle géologique débute donc par la construction d'un modèle à l'échelle la plus grossière, puis par la construction du modèle à l'échelle la plus fine contrainte par le modèle que l'on vient de construire à l'échelle grossière. La boucle de calage reprend alors les étapes précédentes, on envoie ce modèle fin à l'upscaling et en entrée du simulateur d'écoulement, pour obtenir les réponses en production et les comparer aux données réelles.

2.3. Déformation graduelle

La méthode de déformation graduelle est une technique de paramétrisation géostatistique utilisée pour perturber une réalisation d'une variable aléatoire à l'aide d'un nombre réduit de paramètres tout en préservant la variabilité spatiale. Son schéma de base implique la combinaison de deux bruits blancs Gaussiens indépendants :

$$z(\theta) = z_1 \cos(\pi\theta) + z_2 \sin(\pi\theta) \qquad \textbf{Equation 8}$$

z_1 et z_2 sont deux bruits blancs Gaussiens indépendants. Quelle que soit la valeur du paramètre de déformation θ, on peut montrer que z est aussi un bruit blanc Gaussien (Hu, 2000). Comme la règle de déformation est périodique, θ est compris entre -1 et 1. Lorsque $\theta = 0$, z est égal à z_1 ; lorsque $\theta = ½$, z est égal à z_2.

Le bruit blanc Gaussien correspond aux nombres aléatoires attribués aux mailles où on veut simuler des valeurs de la variable. Faire varier le paramètre de déformation permet de varier le bruit blanc Gaussien utilisé pour générer la réalisation. Ce processus permet de déformer la réalisation déduite du bruit blanc Gaussien. Cette technique de paramétrisation, une fois intégrée au processus d'optimisation ou au calage d'historique, est un outil qui permet de parcourir des chaînes de réalisations dans l'espace de recherche. Parmi toutes celles-ci, on peut identifier une réalisation qui permet la décroissance de la fonction objectif. La probabilité d'avoir un bon calage des données dynamiques est très faible si on se contente de parcourir une seule chaine. Pour être efficace, la recherche par déformation graduelle doit se faire sur

plusieurs chaines. Pour plus de détails sur la méthode de déformation graduelle, un ebook a été publié par Le Ravalec *et al.* (2014).

$$z_3(\theta) = z_1 \cdot cos(\pi\theta) + z_2 \cdot sin(\pi\theta)$$

optimisation

$$z_5(\theta) = z_3(\theta_{optim}) \cdot cos(\pi\theta) + z_4 \cdot sin(\pi\theta)$$

optimisation

$$z_7(\theta) = z_5(\theta_{optim}) \cdot cos(\pi\theta) + z_6 \cdot sin(\pi\theta)$$

Figure 8 Schéma d'une chaine d'optimisation à un paramètre de déformation graduelle θ

Pour chaque chaine explorée, on a un processus d'optimisation. Au final, l'optimisation par déformation graduelle correspond à une séquence d'optimisations comme schématisée Figure 8. Une première chaine est construite à partir de deux bruits blancs Gaussiens (z_1 et z_2). On lance un premier processus d'optimisation qui donne la réalisation réduisant le plus la fonction objectif. On obtient donc un θ_{optim} qui donne le bruit blanc Gaussien qui réduit le plus la fonction objectif dans cette première chaine. Comme une chaine seule ne représente qu'une toute petite partie de l'espace de recherche, il faut réitérer le processus. Le bruit blanc Gaussien correspondant à la réalisation optimale du processus d'optimisation précédent est utilisé pour mettre à jour z_1 que l'on appelle ici dans la Figure 8, $z_3(\theta_{optim})$, et un nouveau bruit blanc Gaussien est tiré au hasard pour z_4. La combinaison de ces deux nouveaux bruits blancs Gaussiens donne une nouvelle chaine de réalisations qui une fois parcourue amène à une réalisation optimale qui réduit la fonction objectif autant que possible, et ainsi de suite.

Dans ce contexte, on espère que le schéma multi-échelles apporte plus de flexibilité. Une réalisation dépend de deux bruits blancs Gaussiens : un à l'échelle fine et un à l'échelle grossière. Le bruit blanc Gaussien à l'échelle grossière permet de générer la tendance sur la grille grossière. Le bruit blanc Gaussien à l'échelle fine génère la réalisation sur la grille fine conditionnellement à la réalisation générée précédemment sur la grille grossière. En conséquence, la méthode de déformation graduelle peut être appliquée sur la grille grossière (Figure 9), sur la grille fine (Figure 10) ou sur les deux en même temps. La Figure 9 montre l'impact de la déformation graduelle sur la réalisation simulée sur la grille grossière et les modifications qui en résultent pour la réalisation sur la grille fine. La Figure 10, quant à elle, montre l'impact de la déformation graduelle sur la réalisation à l'échelle fine pour une réalisation grossière fixée. Pour ce dernier exemple, on remarque que la tendance reste la même à l'échelle fine, on joue seulement sur les variations autour de la tendance.

L'avantage de cette méthode de simulation imbriquée est qu'il est possible de changer une réalisation à une échelle donnée, avec un nombre d'inconnues dépendant de cette échelle. Plus l'échelle est grossière, moins il y a de mailles dans la grille associée et donc, moins il y a d'inconnues. Le calage devrait s'en trouver facilité : les données peuvent être intégrées à la bonne échelle avec moins d'inconnues.

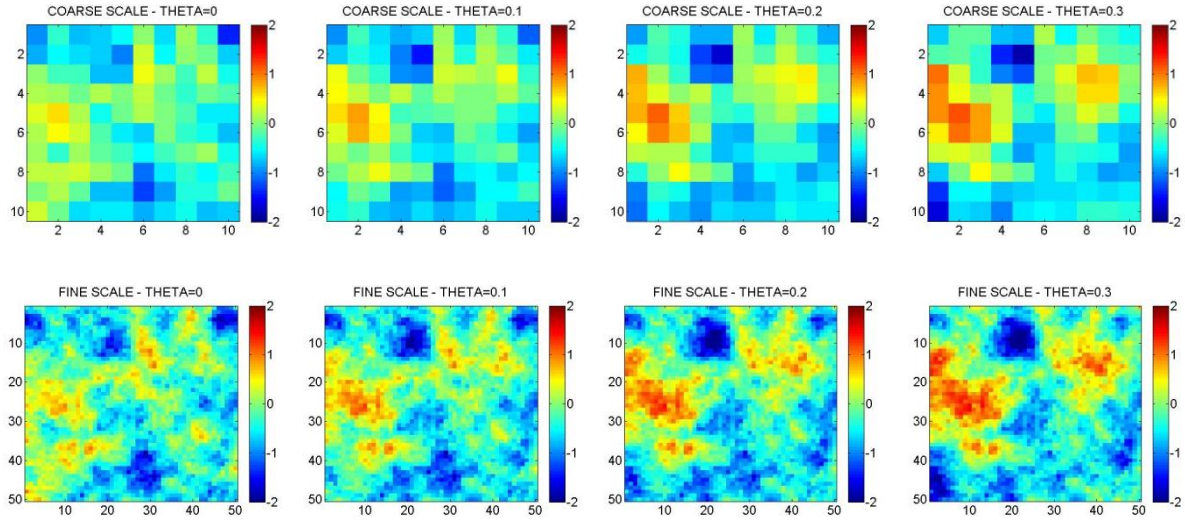


Figure 9. Déformation graduelle d'une réalisation continue à l'échelle grossière. L'équation 6 est appliquée au bruit blanc Gaussien de la grille grossière alors que celui peuplant la grille fine reste inchangé. Le paramètre de déformation graduelle θ est indiqué au-dessus de l'image. Les images du dessus montrent l'impact sur la réalisation grossière. Les images du dessous montrent l'impact sur la réalisation fine.

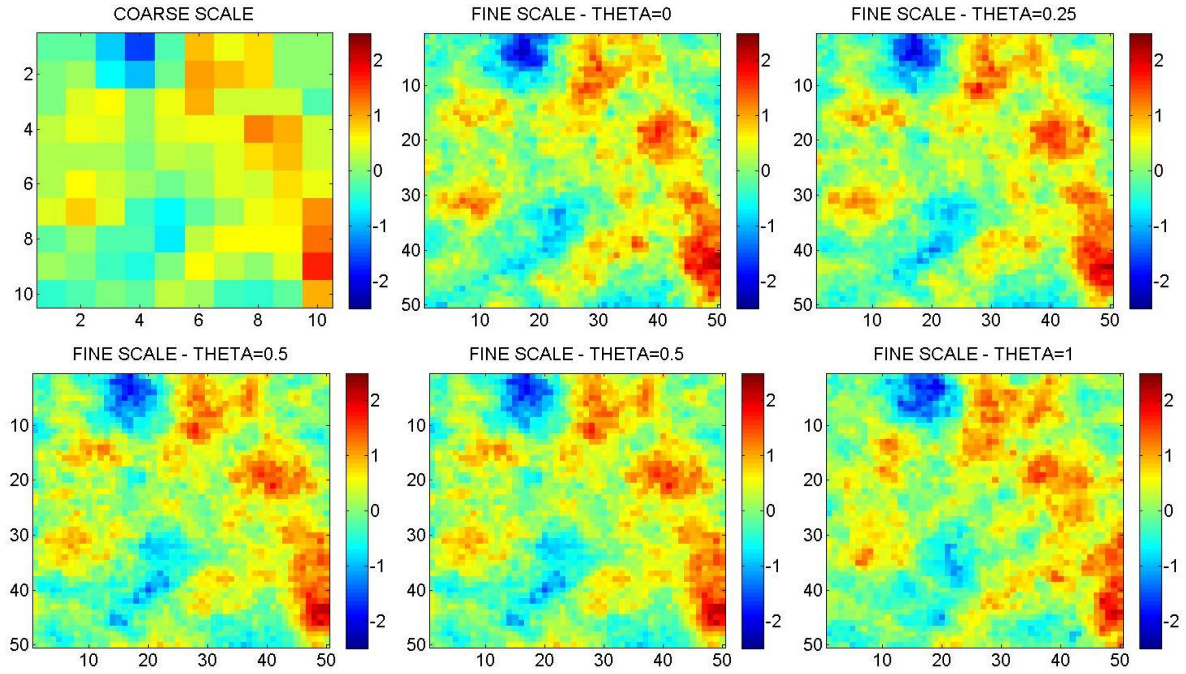


Figure 10. Déformation graduelle d'une réalisation continue à l'échelle fine. L'équation 6 est appliquée ici au bruit blanc Gaussien peuplant la grille fine alors que celui de la grille grossière reste inchangé. Le paramètre de déformation graduelle θ est indiqué sur le dessus de l'image. En haut à gauche : la réalisation grossière ne change pas. Toutes les autres images montrent l'impact sur la grille fine.

Chapitre 3

3. Simulation séquentielle Gaussienne

Dans ce chapitre nous développons un modèle de réservoir issu de la géostatistique à deux points. Nous utilisons donc un variogramme pour modéliser les propriétés pétrophysiques du réservoir. Après une brève bibliographie sur les modèles couramment utilisés, nous présentons la simulation séquentielle Gaussienne, puis nous détaillons l'approche que nous avons proposée pour l'étendre à deux échelles. Nous l'illustrons dans un premier pour des propriétés continues, puis nous expliquons comment procéder pour des variables discrètes. Enfin nous mettons en place un cas synthétique pour mettre en évidence les points fort de la méthode.

3.1. Etat de l'art

Les méthodes géostatistiques sont un ensemble de méthodes statistiques, dédiées à l'analyse de données définies par des coordonnées spatiales et temporelles. Jusqu'à la fin des années 1980, elles étaient essentiellement vues comme un moyen de décrire les variations spatiales à partir d'un variogramme et de prédire les valeurs des propriétés aux points non échantillonnés par krigeage (Vieira *et al*., 1983 ; Trangmar *et al*., 1985). Le krigeage, ou estimation, est un nom générique adopté par la communauté des géostatisticiens pour décrire une famille d'algorithmes de régression par moindre carré. Il existe de nombreuses méthodes de krigeage : simple, ordinaire, universel, ou avec tendance, cokrigeage, krigeage avec dérive externe, *etc*. Une présentation détaillée de ces méthodes mathématiques a été faite par Goovaerts (1997a, pp 125-258).

Le krigeage a tendance à lisser la variabilité spatiale de la propriété modélisée (Goovaerts, 1999). Au contraire, les méthodes de simulation stochastiques ne minimisent pas la variance de l'erreur, mais se concentrent sur la reproduction des propriétés statistiques, notamment la moyenne, la variance et la covariance (ou variogramme). La simulation stochastique est préférée au krigeage pour les applications où la variabilité spatiale de la propriété doit être préservée.

La simulation peut être faite à partir d'une large palette de techniques qui dépendent du modèle de la fonction aléatoire, du nombre et du type d'informations disponibles, et des spécifications sur le temps de calcul. Une revue détaillée est disponible dans le livre de Goovaerts (1997a, pp 376-424), notamment sur les techniques de recuit simulé (Deutsch et Lewis, 1992), d'approche par champ de probabilité (Froidevaux, 1992) et des algorithmes booléens, que nous ne détaillerons pas ici. Nous allons plutôt détailler la simulation séquentielle, méthode dans laquelle s'inscrit la simulation séquentielle Gaussienne.

La simulation séquentielle : au lieu de modéliser la fonction de répartition conditionnelle à *N*-points, on calcule une fonction de répartition conditionnelle à un point sur chaque nœud visité selon une séquence aléatoire. Pour s'assurer de reproduire le variogramme, chaque fonction de répartition conditionnelle est conditionnée non seulement aux données disponibles, mais aussi

aux points déjà simulés. Deux classes se distinguent : les techniques pour la simulation de fonctions aléatoires multi-Gaussiennes ou d'indicatrices.

- La simulation séquentielle Gaussienne (Desbarats, 1996 ; Deutsch et Journel, 1998) : la grille est parcourue séquentiellement. Pour une maille donnée, la moyenne et la variance de la loi conditionnelle (supposée Gaussienne) sont obtenues par krigeage. Cette méthode est détaillée dans la section 3.2
- La simulation séquentielle par indicatrices (Goovaerts, 1997b ; Garcia et Froidevaux, 1997) : la fonction de distribution se détermine à partir du calcul de quelques points de la courbe de distribution (calcul discret de la probabilité pour un certain nombre de seuils convenablement choisis). La simulation non paramétrique utilise des indicatrices.

3.2. Algorithme à une échelle

Nous cherchons à présent à simuler une réalisation d'une variable aléatoire stationnaire $V(\mathbf{u})$, connue en n points u_i, $i \in$ à $[1,n]$, de moyenne m, de variance σ^2 et de covariance $\mathbf{C}^V(\mathbf{h})$. Les n valeurs connues de V sont aussi appelées données dures. Pour ce faire, nous appliquons la méthode de simulation séquentielle Gaussienne, identifiée par l'acronyme SGSim. Les principales étapes de cet algorithme sont présentées ci-dessous.

i) Remplir la grille de départ avec un bruit blanc Gaussien.
ii) Déterminer un chemin aléatoire pour parcourir toute la grille.
iii) Pour chaque maille i visitée, vérifier qu'il n'y a ni donnée dure, ni valeur déjà simulée. Si c'est le cas :

- chercher dans un voisinage proche toutes les données disponibles (les données dures et les valeurs déjà simulées)
- calculer les moyenne m^{SK} et variance $\sigma^{2\,SK}$ de la fonction de densité de probabilité (pdf) en utilisant un krigeage simple :

$$m_i^{SK}(u_i) = m + \sum_{k=1}^{n_i(u_i)} \lambda_k^{SK}(u_i)\left(V(u_k) - m\right)$$
$$\sigma_i^{2\,SK}(u_i) = \sigma^2 - \sum_{k=1}^{n_i(u_i)} \lambda_k^{SK}(u_i) B(u_k) \qquad \textbf{Equation 9}$$

  Les coefficients λ sont les poids du krigeage. Ils sont obtenus par la résolution du système $\mathbf{C}^V \boldsymbol{\lambda} = \mathbf{B}^V$. $\mathbf{C}^V$ est la matrice de covariance calculée pour les mailles avec des données ou des valeurs déjà simulées. $\mathbf{B}^V$ est le vecteur des covariances pour les distances entre la maille i visitée et les mailles voisines contenant des données ou des valeurs déjà simulées.
- à partir de cette pdf et du nombre aléatoire assigné à la maille courante, on obtient une valeur que l'on attribue à la maille i.

iv) Revenir à l'étape iii jusqu'à ce que la grille soit entièrement remplie.

3.3. Algorithme à deux échelles

L'algorithme précédemment décrit permet d'intégrer *n* valeurs connues de la variable *V* en *n* points. Ces données, considérées comme des données dures, sont différentes de celles appelées données molles. Les données dures caractérisent la variable primaire et les données molles une variable secondaire.

Les données dures et les données molles sont des données statiques, mais elles diffèrent de par leur source. Les données dures sont des mesures directes de la propriété *V* à modéliser, appelée variable primaire. Elles correspondent par exemple à des mesures de perméabilité si on cherche à simuler une réalisation d'une variable aléatoire correspondant à la perméabilité. Il n'y en a pas beaucoup, mais elles sont en général précises. Les données molles sont par exemple des données sismiques. Ce ne sont pas des mesures directes de *V*, même si elles donnent des informations sur cette propriété. En général il y a beaucoup de données molles, mais elles ne sont pas très précises. Combiner données dures et données molles peut aider à simuler une réalisation de *V* plus réaliste.

Dans ce contexte, on se concentre sur la simulation de la réalisation de la variable *V*(u), *V* étant caractérisée par une moyenne *m*, une variance σ^2 et une covariance $\mathbf{C}^V$. On suppose que *V* est connue en n_i points u_i, $i \in [1,n_i]$. Ce seront les données dures. Une seconde variable *S*, la variable secondaire, de moyenne $\langle S \rangle$ et de covariance $\mathbf{C}^S$, est connue en n_j points, $j \in [1,n_j]$. Ces données supplémentaires sont considérées comme des données molles. Dans ces conditions, l'algorithme SGSim est très proche de celui introduit dans la section précédente. La seule différence résulte de la définition de la fonction de densité de probabilité calculée au point visité. La moyenne (m^{SCK}) et la variance ($\sigma^{2\ SCK}$) sont calculées par cokrigeage pour tenir également compte des données molles et non plus par krigeage simple.

$$m^{SCK}(u) = m + \sum_{i=1}^{n_i(u)} \mu_i^{SCK}(u)\left(V(u_i) - m\right) + \sum_{j=1}^{n_j(u)} \nu_j^{SCK}(u)\left(S(u_j) - \langle S \rangle\right)$$
$$\sigma^{2^{SCK}}(u) = \sigma^2 - \sum_{k=1}^{n_i(u)+n_j(u)} \lambda_k^{SCK}(u) B(u_k)$$

Equation 10

Le vecteur $\boldsymbol{\lambda}$ contient les poids μ et ν de cokrigeage qui font référence, respectivement, aux données dures et aux données molles. Ces poids sont calculés à partir de la résolution du système $\mathbf{C}\boldsymbol{\lambda}=\mathbf{B}$ avec $\mathbf{C}$ la matrice de covariance et $\mathbf{B}$ le vecteur définis par :

$$C = \begin{pmatrix} C^V_{mn,(m,n)\in[1,n_i(u)]^2} & C^{VS}_{mn,m\in[1,n_i(u)],n\in[1,n_j(u)]} \\ C^{VS}_{mn,m\in[1,n_i(u)],n\in[1,n_j(u)]} & C^S_{mn,(m,n)\in[1,n_{ji}(u)]^2} \end{pmatrix}$$
$$B = \begin{pmatrix} C^V_{um,m\in[1,n_i(u)]} \\ C^S_{un,n\in[1,n_j(u)]} \end{pmatrix}$$

Equation 11

Les matrices de covariance $\mathbf{C}^V$ et $\mathbf{C}^S$ correspondent aux variables *V* et *S*, tandis que $\mathbf{C}^{VS}$ est la matrice de covariance croisée entre ces deux variables.

Dans la section précédente, on a rappelé la méthode classique de simulation séquentielle Gaussienne. L'approche par co-krigeage est lui aussi connu depuis longtemps. Mais dans cette

deuxième section on propose de considéré que *S* est donnée par la moyenne de *V*. De cette façon on cherche à faire apparaître une deuxième échelle, plus grossière. Aussi la simulation stochastique se fait-elle sur deux échelles, une fine et une grossière, en tenant compte d'un lien entre ces deux échelles défini à partir de la moyenne arithmétique.

Il ne s'agit pas ici de proposer une méthode de mise à l'échelle pour construire un modèle de réservoir grossier. On reste dans le domaine de la simulation géostatistique. Le modèle fin ainsi obtenu, même s'il met en œuvre un processus de simulation sur deux échelles, sera ensuite mis à l'échelle suivant une technique appropriée pour obtenir le modèle de réservoir grossier qui sera finalement donné en entrée du simulateur d'écoulement.

3.4. Quelques exemples d'application pour des variables continues

Par souci de simplification, on se concentre sur une variable *V* associée à 2 échelles : une fine et une grossière. La variable *V* est associée à l'échelle fine et donc à la grille fine. On définit une variable V_B comme étant la moyenne de *V* sur une maille grossière de taille/mesure *B*. La variable V_B est associée à l'échelle grossière. On suppose qu'on connait le variogramme de *V*.

3.4.1. Calcul de la covariance à l'échelle grossière et de la covariance croisée

$V(\mathbf{u})$ est supposée connue en n_α points $\mathbf{u}_\alpha$, $\alpha \in [1,n_\alpha]$. On appelle *m* sa moyenne et **C** sa matrice de covariance. La moyenne arithmétique de *V* sur une maille de mesure *B* et centrée en **u** est donnée par :

$$V_B(u) = \frac{1}{|B|}\int_{B(u)} V(u')du' \approx \frac{1}{N_B}\sum_{j,u_j \in B} V(u_j) \qquad \textbf{Equation 12}$$

L'intégrale est approchée par la moyenne de *V* sur les N_B points qui discrétisent la maille grossière choisie. La moyenne de *V* sur la maille de mesure *B* et centrée en $\mathbf{u_k}$ est de façon équivalente notée $V_B(\mathbf{u_k})$ ou V_{Bk}. Comme la somme de variables aléatoires est une variable aléatoire, V_B est aussi une variable aléatoire.
Dans une première étape on détermine la covariance de V_B et la covariance croisée ente *V* et V_B. La covariance croisée est calculée comme suit (pour plus de précision, voir Le Ravalec-Dupin, 2010) :

$$\overline{C_{iBk}} = \frac{1}{N_{Bk}}\sum_{j,u_j \in B_k} C_{ij} \qquad \textbf{Equation 13}$$

Cette covariance représente la moyenne arithmétique des valeurs de la covariance à l'échelle fine calculée pour les distances entre un point $\mathbf{u_i}$ et les N_{Bk} points de la maille grossière centrée en $\mathbf{u_k}$. On peut aussi montrer que la covariance de V_B à l'échelle grossière est égale à :

$$\overline{\overline{C_{BiBj}}} = \frac{1}{N_{Bi}N_{Bj}}\sum_{k,u_k \in Bi}\sum_{l,u_l \in Bj} C_{kl} \qquad \textbf{Equation 14}$$

$\overline{\overline{C_{BiBj}}}$ est la moyenne arithmétique des valeurs de la covariance à l'échelle fine calculées pour les distances entre les N_{BJ} points de la maille grossière centrée en $\mathbf{u_j}$ et les N_{Bi} points de la maille

grossière centrée en $\mathbf{u}_i$. La Figure 11 montre la covariance calculée à l'échelle grossière en supposant que la covariance à l'échelle fine est exponentielle avec une portée de 20 unités. On peut voir, comme on s'y attendait, que plus la grille est grossière et plus la variance (*i.e.*, la covariance pour une distance nulle) est petite. En outre, les variations spatiales de V_B s'en trouvent d'autant plus lissées : le comportement de la covariance de V pour l'échelle fine est linéaire à l'origine alors que celui mis en évidence à l'échelle grossière tend à être parabolique.

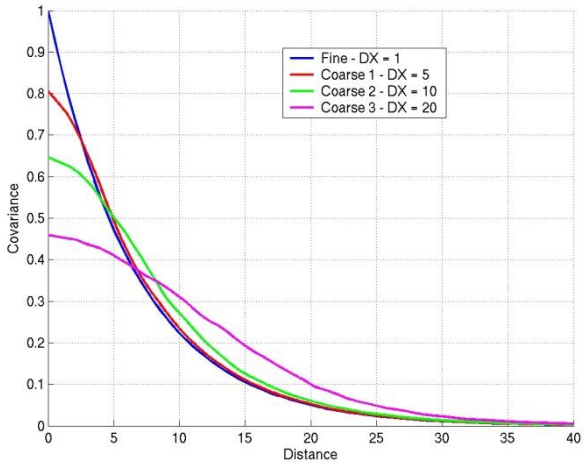


Figure 11. Covariances calculées à l'échelle grossière connaissant la covariance à l'échelle fine. L'échelle fine est associée à une grille de taille de maille l'unité. La covariance à l'échelle fine est exponentielle et de portée égale à 20 unités. Trois grilles grossières sont considérées avec des tailles de mailles respectivement de 5, 10 et 20 unités.

3.4.2. Exemples d'applications

Toutes les covariances étant à présent connues (covariance à l'échelle fine, covariance à l'échelle grossière, covariance croisée), l'algorithme usuel de simulation séquentielle Gaussienne peut être étendu comme suit. Premièrement on simule une réalisation de V_B à l'échelle grossière comme expliqué dans la section 3.2. Puis, on considère que la réalisation à l'échelle grossière est une donnée molle. L'étape suivante consiste à simuler une réalisation de V à l'échelle fine conditionnée par la réalisation simulée au préalable à l'échelle grossière. L'algorithme est explicité dans la section 3.3. Clairement, la moyenne de V_B est la même que la moyenne de V.

La Figure 12 illustre ce processus de simulation à deux échelles. La première réalisation obtenue à l'échelle grossière est montrée en haut, à gauche. Ensuite, trois autres réalisations sont simulées à l'échelle fine tout en étant conditionnées par la réalisation à l'échelle grossière. La seule différence entre les trois réalisations est le germe utilisé pour initialiser le processus aléatoire. On peut voir que les simulations à l'échelle fine respectent la tendance donnée par la réalisation à l'échelle grossière.

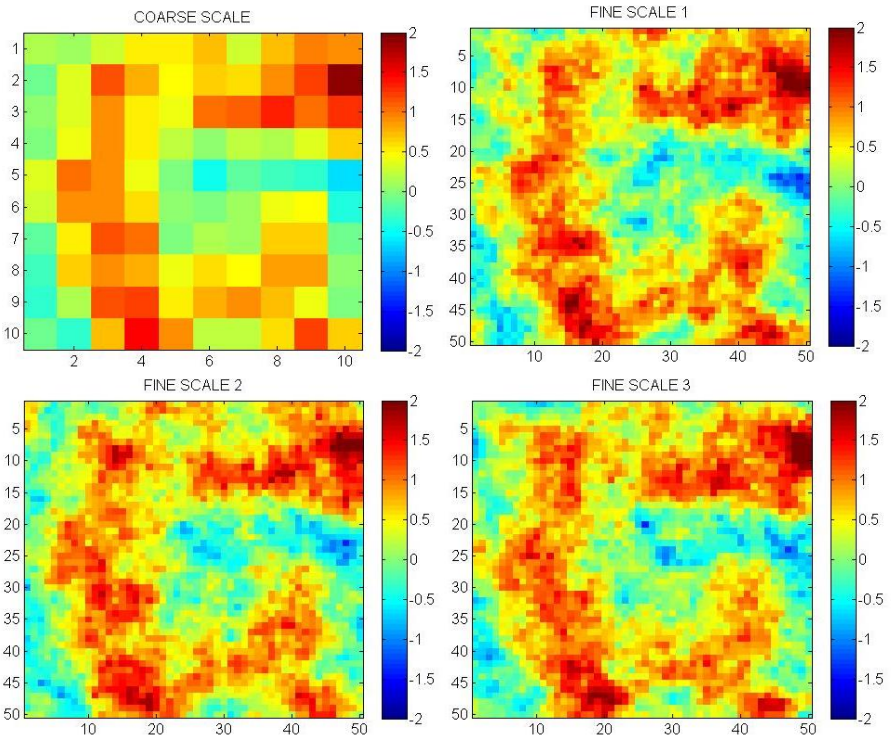


Figure 12. En haut à gauche : réalisation simulée à l'échelle grossière. En haut à droite et en dessous : réalisations simulées à l'échelle fine conditionnellement à la réalisation à l'échelle grossière. A l'échelle fine *V* est caractérisée par une moyenne nulle, une variance égale à 1 et une covariance exponentielle de portée 15 unités. La taille des mailles est de 1 unité à l'échelle fine et de 5 unités à l'échelle grossière.

3.5. Simulation multi-échelles pour des variables discrètes

La méthode de simulation multi-échelles introduite ci-dessus pourrait être étendue à des variables discrètes en utilisant la méthode de simulation séquentielle par indicatrices (Goovaerts, 1997). L'alternative développée dans le cadre de ce travail s'appuie sur la combinaison de la méthode de simulation séquentielle Gaussienne et de la méthode des gaussiennes tronquées (Chilès et Delfiner, 1999).

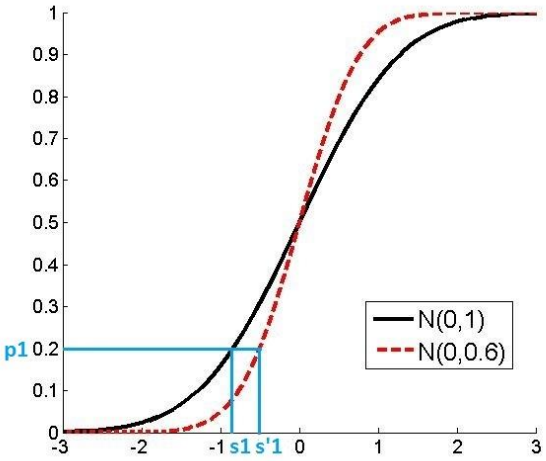


Figure 13. Estimation des seuils de troncature déduits des proportions de faciès. Pour cela il faut inverser la fonction de densité de probabilité cumulée de la variable continue à tronquer. A l'échelle fine, le seuil associé à p_1 (proportion de faciès 1) est s_1 tandis que à l'échelle grossière il devient s'_1.

La méthode des Gaussiennes tronquées consiste à tronquer une variable Gaussienne continue à l'aide de seuils préalablement calculés à partir des proportions des différentes classes de la variable discrète. Considérons la simulation d'une réalisation comportant 3 faciès, notés F1, F2 et F3, de proportions p_1, p_2 et p_3. On génère dans un premier temps une réalisation continue de moyenne nulle, de variance 1 et de covariance C. Puis, on déduit les valeurs des seuils de la formule suivante :

$$s_i = G^{-1}\left(\sum_{k=1}^{i} p_k\right) \qquad \textbf{Equation 15}$$

où G représente la fonction de répartition de la loi normale standard et i le faciès considéré. L'exemple de la Figure 10 montre le lien entre la proportion p_1 de faciès F1 et le seuil s_1. Les seuils étant connus, la réalisation continue est transformée en une réalisation discrète : les valeurs se trouvant entre deux seuils s_{i-1} et s_i sont converties en indicateur i, qui représente le faciès Fi (Figure 14).

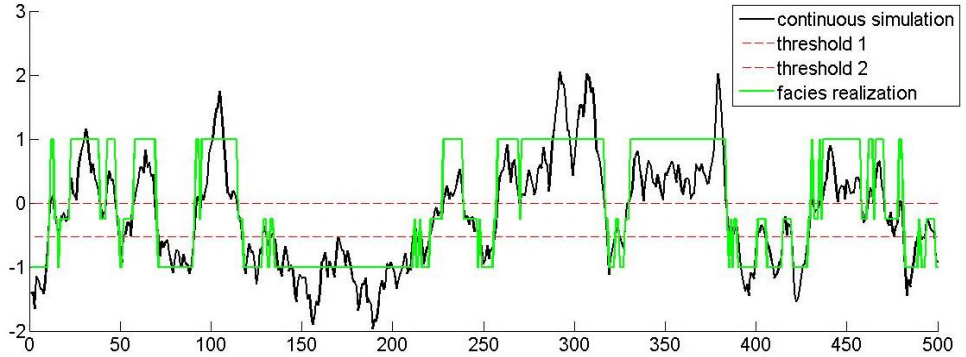


Figure 14. Simulation 1D d'une réalisation de 3 faciès par la méthode des Gaussiennes tronquées. La courbe verte montre la réalisation en faciès alors que la courbe noire montre la réalisation continue normale standard sous-jacente. Les proportions des faciès F1, F2 et F3 sont respectivement de 30%, 20% and 50%. La covariance de la variable continue est exponentielle de portée 25 unités.

Passer à une méthode multi-échelles ne nécessite que peu de changements. Encore une fois, pour plus de simplicité, on se restreint à deux échelles.
Pour la simulation de la réalisation continue sous-jacente, il suffit d'appliquer l'algorithme de simulation séquentielle Gaussienne multi-échelles décrit précédemment. Autrement dit, on génère une réalisation continue de V_B à l'échelle grossière, puis on simule une réalisation continue de V à l'échelle fine conditionnellement à la réalisation à l'échelle grossière. V est une variable Gaussienne normale standard, sa moyenne est nulle, sa variance vaut 1 et sa covariance est C.

Le variogramme de V étant connu, on peut caractériser V_B : sa moyenne est nulle, sa variance est égale à $\overline{\overline{C_{BiBj}}}(0)$ et sa covariance à $\overline{\overline{C_{BiBj}}}(h)$ (voir Equation 14).

L'étape suivante consiste à convertir les deux réalisations continues (fine et grossière) en réalisations discrètes. Pour cela, on a besoin des seuils de troncature. Les proportions des faciès sont les mêmes pour toutes les échelles. A l'échelle fine, les seuils peuvent être calculés à partir de l'Equation 15 puisque la variable sous-jacente suit une loi normale standard. A l'échelle grossière, V_B suit aussi une loi normale, mais de variance inférieure à 1 à cause de la prise de moyenne. Aussi, pour l'échelle grossière, les seuils sont-ils calculés à partir de l'inverse

de la fonction de répartition de la loi normale de moyenne nulle et de variance $\overline{\overline{C_{BiBj}}}(0)$ (Figure 10). A l'échelle fine, le seuil s_1 associé à la proportion p_1 de faciès F1 est calculé à partir de l'inverse de la fonction de répartition de la loi *N(0,1)*. A l'échelle grossière, la variance de V_B est égale à 0.6 pour le cas considéré. Dans ces conditions, le seuil associé à la proportion p_1 de faciès F1 n'est plus s_1, mais s'_1. Cette procédure assure l'équivalence des proportions à l'échelle fine et à l'échelle grossière.

Un exemple de simulation à deux échelles est donné sur la Figure 15. La réalisation à l'échelle grossière est en haut, à gauche. Trois réalisations en faciès différentes sont générées à l'échelle fine, conditionnellement à la réalisation à l'échelle grossière. Encore une fois, on peut voir que les réalisations à l'échelle fine respectent la tendance donnée par la réalisation à l'échelle grossière.

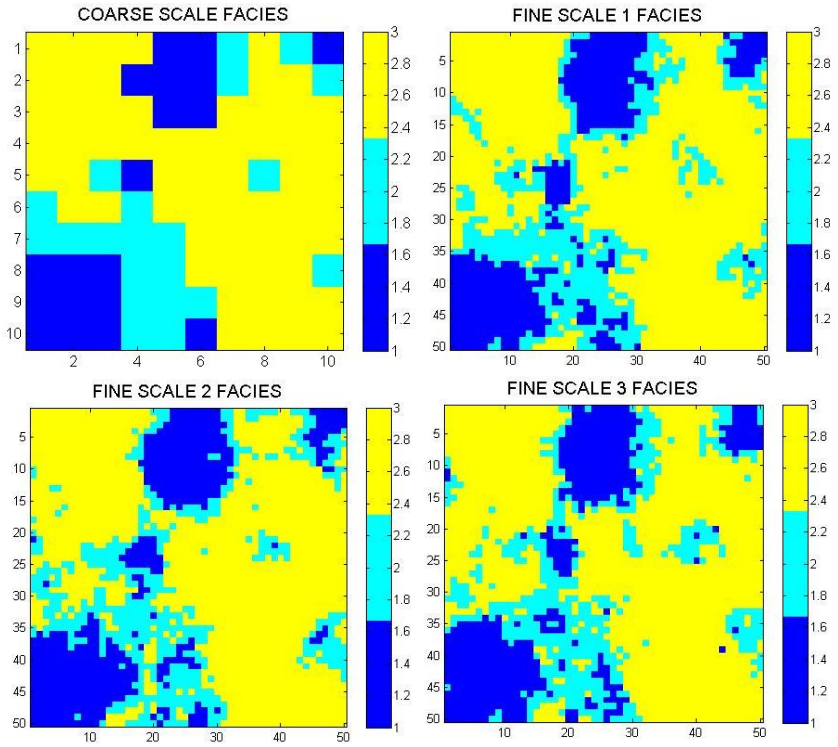


Figure 15. En haut à gauche : réalisation discrète simulée à l'échelle grossière. En haut à droite et en dessous : réalisations discrètes simulées à l'échelle fine conditionnées par la réalisation à l'échelle grossière. La covariance de la variable normale standard à l'échelle fine est exponentielle et de portée 150 unités. Il y a 3 faciès de proportions 30%, 20% et 50%. La taille d'une maille à l'échelle fine est de 10 unités et à l'échelle grossière de 50 unités.

3.6. Exemple de calage multi-échelles

Cette section vise à mettre en avant le potentiel de la technique de paramétrisation multi-échelles à partir de l'étude d'un cas synthétique. On cherche à construire un modèle de réservoir respectant à la fois des données de production et des données de saturation. Ces dernières peuvent être plus ou moins comparées à des données sismiques au sens où elles représentent une information couvrant tout le réservoir.

Dans un premier temps, on construit un modèle de réservoir de référence. Ce modèle est donné en entrée à un simulateur d'écoulement pour définir les données de référence (débits d'huile et d'eau, pressions au puits injecteur, et grilles de saturation). Dans un deuxième temps,

on suppose le modèle de réservoir de référence inconnu. On met alors en place un processus de calage basé sur la méthode de déformation graduelle afin de déterminer un modèle de réservoir reproduisant les données de référence.

3.6.1. Cas de référence

Le modèle de perméabilité de référence (Figure 16, en haut à gauche) est généré en utilisant l'algorithme standard de simulation séquentielle Gaussienne. Il est constitué d'une grille de 50×50 mailles, chacune de dimension 10×10 m². Le logarithme de la perméabilité est caractérisé par une moyenne de 1.5, une variance de 0.8 et un variogramme exponentiel de portée 150 m, donné par :

$$\gamma(h) = \sigma^2 \left[1 - \exp\left(-3 \frac{h}{l_c} \right) \right] \qquad \textbf{Equation 16}$$

h représente la distance entre 2 points, l_c la portée et σ^2 la variance. La perméabilité est définie en millidarcy (mD). Par souci de simplification, la porosité est fixée à 0,2 partout. Le réservoir est initialement saturé en huile. La production est basée sur une injection d'eau. Un puits injecteur est situé dans la maille (1;1) et un puits producteur à l'opposé dans la maille (50;50). On suppose qu'on connait la perméabilité aux puits : $\log_{10}(k)$ est égale à 1,6015 mD à l'injecteur et 1,2795 mD au producteur. Les 8000 jours d'historique de production sont simulés à partir d'un simulateur black-oil à deux phases. Le débit d'injection d'eau est fixé à 100m^3/jour et la pression au puits producteur à 5000 kPa. On obtient alors un ensemble de données de référence : la pression à l'injecteur (Figure 16, en haut à droite), les débits d'eau et d'huile au producteur (Figure 16, en bas à gauche) et la saturation dans tout le réservoir après 1500 jours et 2700 jours d'injection (Figure 16, en bas à droite).

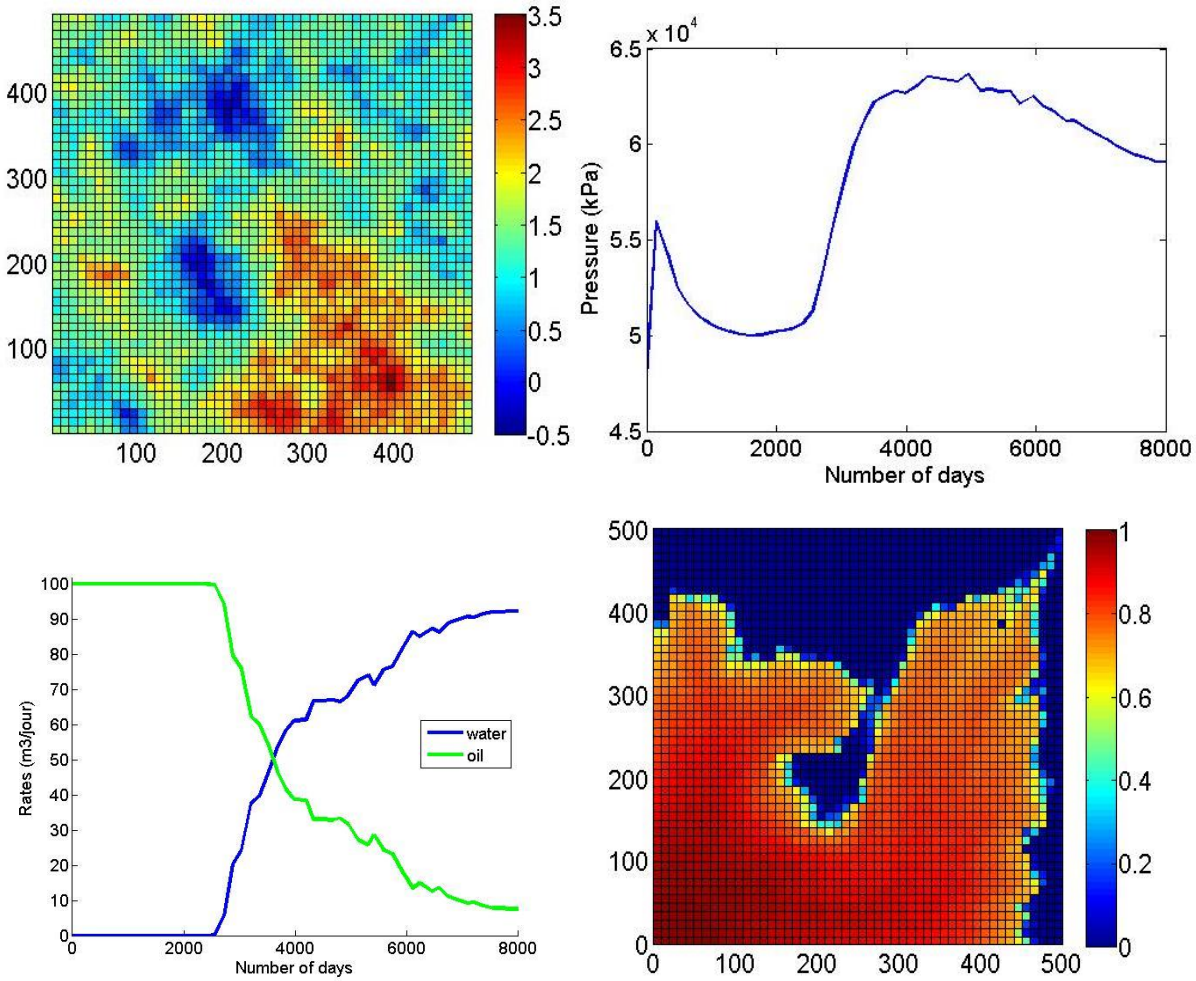


Figure 16. En haut à gauche : champ de log-perméabilité de référence – l'unité de la perméabilité est le mD. En haut à droite : pression au cours du temps à l'injecteur. En bas à gauche : en bleu débit d'eau au producteur et en vert débit d'huile. En bas à droite : saturation en haut sur tout le réservoir après 2700 jours d'injection.

3.6.2. Schéma de calage

A ce stade, le champ de référence de la perméabilité est supposé inconnu. Le but est de générer un modèle de perméabilité initial et de l'ajuster en utilisant la méthode de déformation graduelle pour qu'il reproduise les données de référence.

Le schéma de calage est décrit sur la Figure 7 (section 2.2). La première étape est la simulation d'un modèle de perméabilité. Il s'agit de simuler une réalisation du logarithme de la perméabilité à l'échelle grossière, puis, de simuler une réalisation du logarithme de la perméabilité à l'échelle fine conditionnellement à la réalisation déjà simulée à l'échelle grossière. La deuxième étape consiste à donner ce modèle de perméabilité fin au simulateur d'écoulement pour calculer les réponses numériques requises pour le calage : les pressions à l'injecteur, les débits d'huile et d'eau au producteur et les saturations sur tout le réservoir aux temps 1500 jours et 2700 jours. La troisième étape est le calcul de la fonction objectif qui mesure l'écart entre les données de référence et les réponses simulées :

$$J(\mathbf{v}) = w_p \frac{1}{N_p} \sum_{i=1}^{N_p} \left(\frac{g(\mathbf{v}) - d_p}{\sigma_p} \right)^2 + w_s \frac{1}{N_s} \sum_{i=1}^{N_s} \left(\frac{g(\mathbf{v}) - d_s}{\sigma_s} \right)^2 \quad \textbf{Equation 17}$$

Le premier terme mesure l'écart par rapport aux données de production : il dépend des données à l'injecteur et au producteur. Le second terme mesure l'écart par rapport aux données de saturation : il est relié aux grilles de saturation. Le vecteur **v** représente le modèle de perméabilité et l'opérateur *g* le simulateur d'écoulement. *N* est le nombre de séries de données et **d** le vecteur des données de référence. σ est la déviation standard des erreurs de données. Les indices *p* et *s* font référence, respectivement, aux réponses en production et en saturation. Les coefficients *w* sont des poids utilisés pour normaliser la contribution des deux termes d'erreur. Ils sont choisis de façon à initialiser la fonction objectif à 2 : 1 pour le terme d'erreur sur la production et 1 pour le terme d'erreur sur la saturation.

Le but du processus de calage est de déterminer **v** tel que la fonction objectif soit aussi petite que possible. Comme expliqué ci-dessus, **v** est le modèle de perméabilité fin. Il dépend des bruits blancs Gaussiens générés à l'échelle fine et à l'échelle grossière. La grille fine est la grille 50×50 utilisée pour construire le modèle de référence. La grille grossière est une grille de 10×10 mailles, chacune de dimension 50×50 m².

Dans ce cas d'étude, on combine le processus de minimisation et la méthode de déformation graduelle et on examine trois cas :

1) on applique la méthode de déformation graduelle pour perturber la réalisation à l'échelle grossière seulement ;
2) on applique la méthode de déformation graduelle pour perturber la réalisation à l'échelle fine seulement ;
3) on applique la méthode de déformation graduelle d'abord à l'échelle grossière, puis à l'échelle fine.

Dans tous les cas, le processus de minimisation est contrôlé par un unique paramètre de déformation.

3.6.3. Résultats

3.6.3.1 Test 1: déformation à l'échelle grossière

Dans un premier temps, on procède à trois calages (comme décrit dans la Figure 7 section 2.2) en partant d'un même modèle initial de perméabilité et en appliquant la méthode de déformation graduelle au bruit blanc Gaussien de la grille grossière. Chaque processus de calage comprend 10 optimisations successives visant à investiguer ainsi 10 chaînes de réalisations. Chaque processus d'optimisation nécessite entre 8 et 10 simulations d'écoulement. Les différences entre les 3 tests découlent uniquement des bruits blancs Gaussiens (z_2) générés au début de chaque processus d'optimisation pour faire une combinaison graduelle (Equation 8). La Figure 17 montre l'évolution de la fonction objectif au cours des itérations, chaque itération nécessitant une simulation d'écoulement. Elle décroît de 30 à 60% suivant le cas considéré.

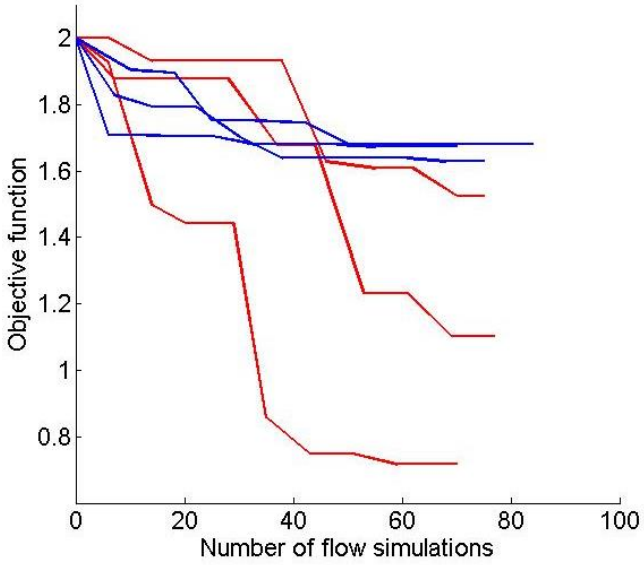


Figure 17. Évolution de la fonction objectif en fonction du nombre de simulations d'écoulement. En rouge : optimisation avec déformation graduelle à l'échelle grossière. En bleu : optimisation avec déformation graduelle à l'échelle fine.

On répète le même test en partant à présent d'un modèle de perméabilité initial plus éloigné du cas de référence. Les résultats obtenus sont montrés sur la Figure 18. Une fois encore, on observe que la déformation à l'échelle grossière permet de décroître significativement la fonction objectif.

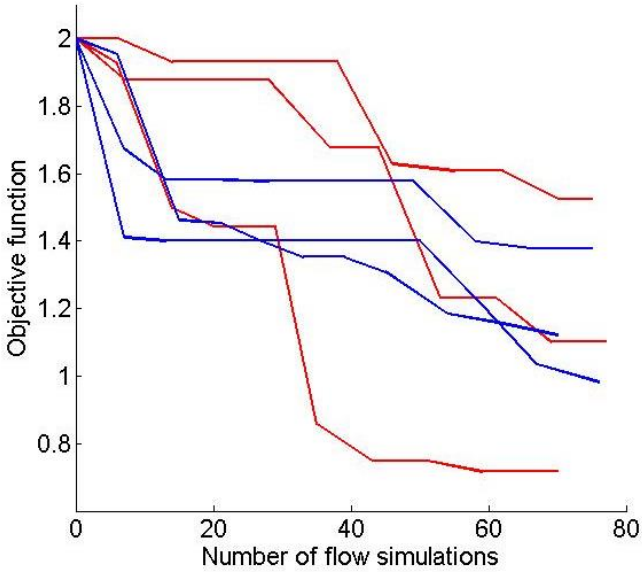


Figure 18. Évolution de la fonction en fonction du nombre de simulations d'écoulement avec déformation graduelle à l'échelle grossière seulement. En rouge : le modèle de perméabilité initial est plutôt proche du cas de référence. En bleu : le modèle de perméabilité initial est plus éloigné du cas de référence.

Le meilleur modèle de perméabilité obtenu au cours des tests ainsi que les réponses en écoulement calculées pour ce modèle sont montrés Figure 19. Ces réponses respectent assez bien les données de référence.

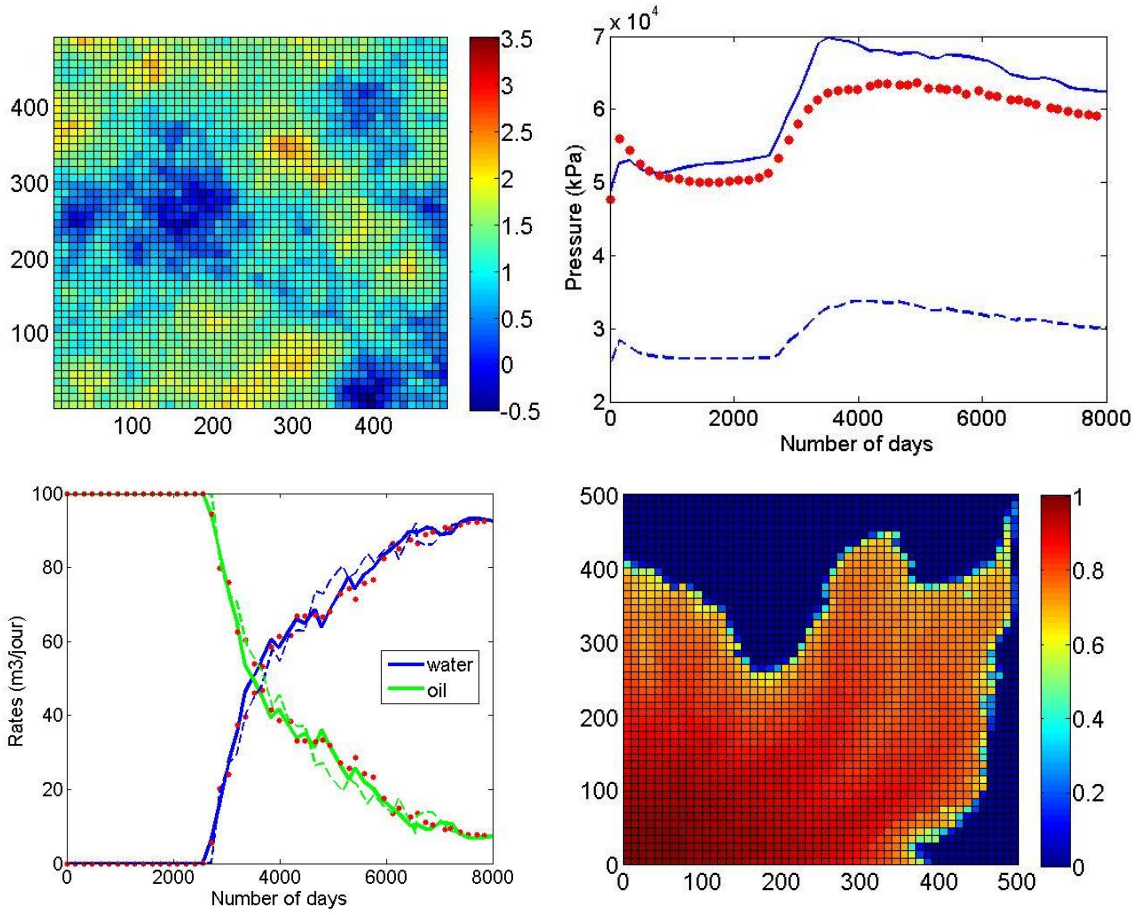


Figure 19. En haut à gauche : meilleur modèle de log-perméabilité obtenu – la perméabilité est en mD. En haut à droite : évolution de la pression à l'injecteur au cours du temps. En bas à gauche : débits d'huile (vert) et d'eau (bleu) au producteur au cours du temps. En bas à droite : carte de saturation en eau sur tout le réservoir au bout de 2700 jours. Points rouges : données de référence. Lignes pointillées : réponses simulées pour le modèle initial. Lignes pleines : réponses simulées pour le modèle en haut à gauche.

3.6.3.2 Test 2 : déformation à l'échelle fine

Dans un deuxième temps, on applique la méthode de déformation graduelle à l'échelle fine. On répète le processus de calage trois fois en partant du même modèle de perméabilité initial, mais en tirant à chaque processus d'optimisation des bruits blancs Gaussiens z_2 différents pour l'échelle fine. Le modèle de perméabilité grossier reste le même tout le long des tests. La Figure 17 montre la décroissance de la fonction objectif : la fonction diminue beaucoup moins que lorsque la déformation graduelle est appliquée à l'échelle grossière. Au mieux, la fonction objectif décroît de 27%.

3.6.3.3 Test 3: déformation à l'échelle grossière, puis à l'échelle fine

Les résultats obtenus précédemment mettent en évidence que la déformation à l'échelle grossière est plus efficace que la déformation à l'échelle fine, au moins pour le cas étudié. On examine maintenant le potentiel d'une approche avec déformation d'abord à l'échelle grossière, puis à l'échelle fine. On revient au modèle de perméabilité initial qui permet la décroissance la plus importante et on lance le processus de calage trois fois. On observe que la déformation à l'échelle fine au cours du second processus d'optimisation ne permet pas de réduire beaucoup plus la fonction objectif.

3.6.3.4 Test 4: influence de la taille de la grille grossière

Dans un dernier test, on va chercher à comprendre l'influence de la taille de maille de la grille grossière sur le processus de calage. Les différentes grilles grossières considérées sont listées ci-après :
- une grille grossière avec 5×5 mailles de dimension 100×100 m^2;
- une grille grossière avec 10×10 mailles de dimension 50×50 m^2;
- une grille grossière avec 20×20 mailles de dimension 25×25 m^2; et
- une grille grossière avec 30×30 mailles de dimension 16.67×16.67 m^2.

La grille 10×10 est celle que l'on a prise pour les différents tests préalablement présentés. Encore une fois, on veut déterminer le modèle de perméabilité qui respecte toutes les données de référence, en appliquant la méthode de déformation graduelle à l'échelle grossière, l'échelle grossière correspondant aux grilles ci-dessus. L'évolution de la fonction objectif est montrée sur la Figure 20. On voit que plus la grille grossière est raffinée entre 5×5 et 20×20, plus la fonction objectif diminue. Toutefois, pour la grille 30×30, la diminution de la fonction objectif semble gênée. Au début, l'augmentation de la résolution de la grille grossière rend le processus de calage plus efficace, en augmentant sa flexibilité. Cependant, à partir d'un certain point, il y a trop d'inconnues pour un seul paramètre de déformation.

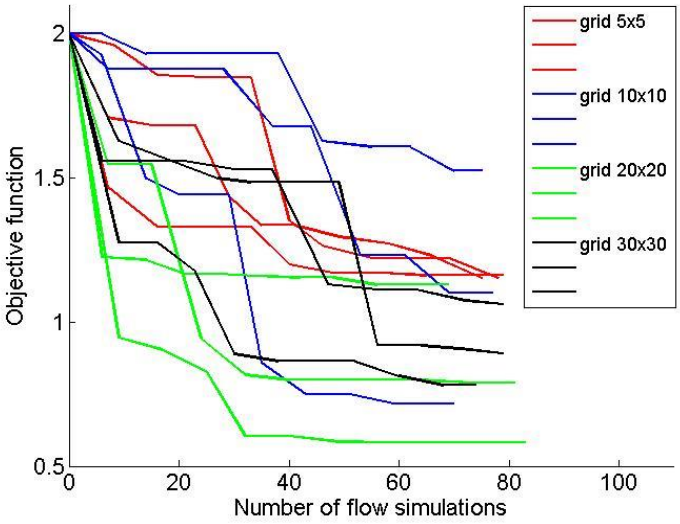


Figure 20. Évolution de la fonction objectif selon le nombre de simulations d'écoulement lors du processus d'optimisation avec déformation graduelle à l'échelle grossière. On teste l'influence de la résolution de la grille grossière, *i.e.*, les grilles ont 5×5, 10×10, 20×20 et 30×30 mailles.

3.7. Comparaison d'un calage avec SGSim classique et du calage 2 échelles avec déformation sur l'échelle grossière

Dans ce paragraphe on va comparer le calage avec SGSim classique à une seule échelle avec les résultats obtenus pour le calage à deux échelles.

On commence par faire le calage avec SGSim classique. On prend le même cas de référence que précédemment (décrit dans le paragraphe 3.6.1) :
- Grille de 50×50 mailles, chacune de dimension 10×10 m²
- Le logarithme de la perméabilité est caractérisé par une moyenne de 1.5, une variance de 0.8 et un variogramme exponentiel de portée 150 m
- La porosité est fixée à 0.2 partout

- Le réservoir est initialement entièrement saturé en huile, il y a un injecteur dans la maille (1,1) et un producteur dans la maille (50,50)
- On connaît le logarithme de la perméabilité aux puits qui est de 1.6015 mD à l'injecteur et 1.2795mD au producteur

On simule sur 8000 jours avec comme contrainte, un débit d'injection d'eau de 100m^3/jour et une pression à l'injecteur de 5000 kPa.
Ce sont les données que l'on obtient que l'on compare aux données de références.
Les étapes du calage sont donc :

1) Simulation du réservoir avec l'algorithme classique SGSim : on obtient une carte de perméabilité
2) La carte de perméabilité est envoyée dans le simulateur d'écoulement
3) Les réponses en pression, en débit et les cartes de saturation sont comparées aux données de références décrites dans le paragraphe 3.6.1
4) On perturbe le bruit blanc gaussien associé à l'unique grille et on revient à l'étape 1 jusqu'à avoir fait 10 itérations de calage

On compare ensuite l'évolution de la fonction objectif par rapport au calage du modèle multi-échelles avec déformation sur la grille grossière uniquement qui est le plus efficace.

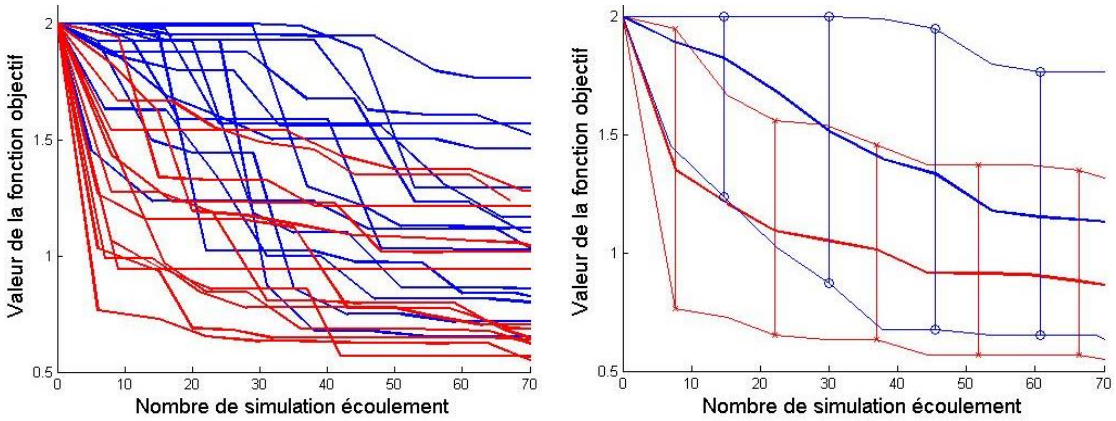


Figure 21 Comparaison des calages sur le cas de référence. A gauche. Les courbes en bleu représentent les fonctions objectifs au cours des calages pour le calage multi-échelles en ne calant que la grille grossière. Les courbes en rouges représentent les fonctions objectifs pour le calage mono-échelle avec un SGSim classique. A droite. Les courbes en plein et gras représentent la moyenne des fonctions objectifs, en bleu pour le calage multi-échelles, en rouge pour le calage mono-échelle. Les courbes enveloppes représentent le minimum et le maximum obtenus lors des différents calages, en bleu pour le multi-échelles, en rouge pour le mono-échelle.

Ce dernier test ne permet pas vraiment de conclure sur le bénéfice ou non de la méthode multi-échelles dans le calage d'historique par rapport à la méthode mono-échelle. Sur 10 cycles d'optimisation, les deux méthodes présentent des cas où le calage marche bien et des cas où le calage ne marche pas vraiment. Les raisons étant qu'il n'y a qu'un seul paramètre de déformation graduelle et seulement 10 cycles d'optimisation. Pour conclure, il faut d'abord optimiser le code pour que le temps de simulation soit raisonnable et ensuite faire un cas qui soit plus réaliste. Cependant il est certain que ces résultats dépendront du cas choisi, s'il se prête ou non à une décomposition en plusieurs échelles.

3.8. Cas de propriétés discontinues : faciès

A partir du même modèle que précédemment (même modèle de variogramme, mêmes dimensions et même résolution de grille, même pression au producteur et même débit à

l'injecteur), on construit un modèle en faciès. Le seul changement est que la réalisation continue à l'échelle fine suit une loi normale standard. On suppose que le réservoir comprend 3 faciès F1, F2 et F3, de proportions 30, 20 et 50%. Les perméabilités associées à ces faciès sont constantes et valent 0,5, 3,2 et 1,8 mD. Le modèle en faciès de référence est montré sur la Figure 22, en haut, à gauche. Le simulateur d'écoulement donne pour ce modèle de référence un ensemble de données de production de référence : pressions à l'injecteur (Figure 22, en haut à droite), débits d'huile et d'eau au producteur (Figure 22, en bas à gauche) et carte de saturation en eau à 2700 jours (Figure 22, en bas à droite).

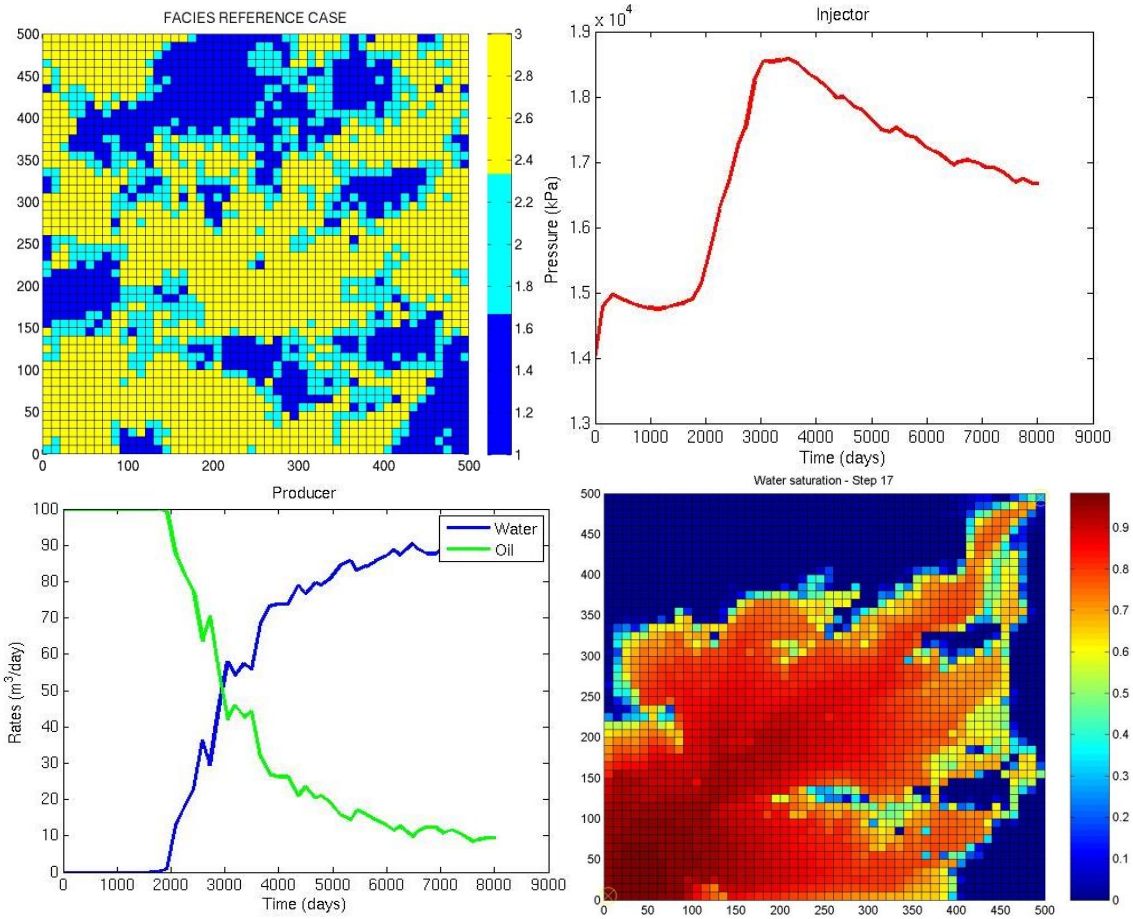


Figure 22. Cas de référence. En haut à gauche modèle en faciès. En haut à droite : pression à l'injecteur au cours du temps. En bas à gauche : débits d'huile et d'eau au producteur au cours du temps. En bas à droite : carte de saturation en eau à 2700 jours.

On garde le même schéma de calage que celui décrit dans la section 3.6.2 ci-dessus. On donne ici les résultats du meilleur calage.

On montre seulement la déformation à l'échelle grossière, cette approche étant celle qui fournit les meilleurs résultats. Les résultats sont similaires à ceux obtenus pour une propriété continue. La fonction objectif décroit de 40 à 60% environ.

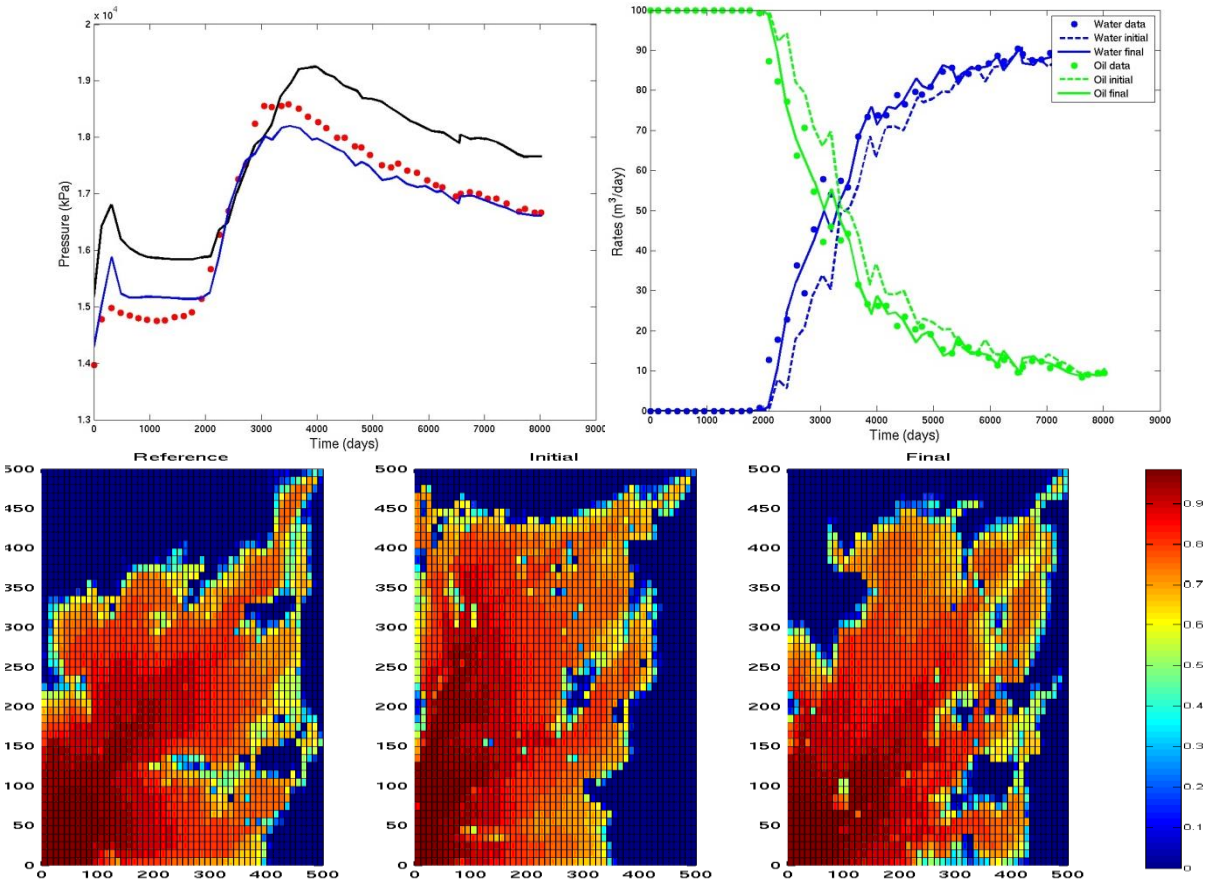


Figure 23. Meilleur cas d'optimisation. En haut à gauche : Pression à l'injecteur, en noir la pression initiale, en bleu la pression finale et en rouge les données. En haut à droite : les débits d'huile et d'eau au producteur. En vert : l'huile, en bleu : l'eau. En pointillé : les valeurs initiales, en plein : les valeurs optimales et en point : les données de référence. En bas, de gauche à droite, saturation en eau à 2700 jours pour le modèle de référence, le modèle initial et le modèle final.

3.9. Conclusion

On a développé, dans ce chapitre, un algorithme de modélisation de réservoir multi-échelles, basé sur la simulation séquentielle Gaussienne. Puis on a paramétrisé la méthode pour le calage d'historique de production, de façon à ce que les échelles puissent être modifiées indépendamment les unes des autres. La méthode de perturbation choisie est la déformation graduelle, car elle a l'avantage de conserver la variabilité spatiale de la propriété modélisée. Enfin on a testé la méthode sur un cas synthétique.

Pour simplifier notre approche, on a choisi de considérer 2 échelles uniquement, une grossière et une fine. Cependant, l'extension de la méthode à plusieurs échelles est directe. Le but de la technique de simulation multi-échelles est de générer une réalisation d'une variable aléatoire V à l'échelle fine ainsi que sa moyenne à n'importe quelle échelle plus grossière. Tout d'abord, on s'est concentré sur le calcul de la covariance de la variable moyennée et de la covariance croisée à partir de la variable V à l'échelle fine. Ensuite, on a proposé de suivre le schéma suivant : on génère une première réalisation de la variable moyenne à l'échelle grossière, puis on simule la réalisation de V à l'échelle fine conditionnellement à la réalisation à l'échelle grossière. Après avoir considéré le cas des propriétés continues, nous avons étendu la méthode aux propriétés discrètes.

Puis, nous avons montré comment combiner la méthode de simulation multi-échelles proposée avec la méthode de déformation graduelle, le processus de déformation pouvant être effectué aussi bien à l'échelle grossière qu'à l'échelle fine. En d'autres termes, on peut varier la moyenne de la variable simulée à l'échelle grossière ou ses fluctuations autour de la moyenne à l'échelle fine.

Finalement, nous avons construit un cas synthétique pour apprécier le potentiel de la méthode proposée. De nombreux tests nous ont permis de voir que le processus de calage est plus efficace lorsqu'on applique la déformation graduelle à l'échelle grossière plutôt qu'à l'échelle fine.

L'idée derrière le développement de cet algorithme est de copier l'architecture du réservoir et de prendre en compte les différentes échelles de sa structure et des données qui y sont associées. Les grosses hétérogénéités sont représentées à l'échelle grossière (basse résolution) et les plus petites à l'échelle fine (haute résolution). Les données sismiques sont basse résolution comparées aux données de puits qui elles sont très haute résolution. Cette approche devrait rendre le calage d'historique de production plus flexible en le divisant en deux parties. Dans un premier temps, l'échelle grossière est calée pour obtenir la structure générale du réservoir. Puis dans un deuxième temps le calage de l'échelle fine permet d'ajuster le modèle. C'est cette flexibilité qui nous intéresse pour le calage d'historique : la réalisation à l'échelle grossière inclut moins de mailles et donc moins d'inconnues. Ce degré de flexibilité nous permet d'envisager une paramétrisation économique pour le calage d'historique de production. Dans le cas test, on voit que le calage est flexible, les échelles peuvent être modifiées indépendamment. Cependant sur ce cas la comparaison avec le calage mono-échelle n'est pas vraiment concluant. D'autres cas test plus complexes, plus réalistes sont à effectués. De plus il serait intéressant d'étudier l'intégration des données selon leur échelle, pour savoir s'il y aurait un impact sur le calage.

La méthode de simulation stochastique utilisée s'inscrit dans le cadre d'une statistique à deux points : elle repose sur un variogramme qui rend compte des corrélations entre deux points du réservoir selon la distance qui les sépare. Le recours au variogramme est pratique et simple. Toutefois, il ne permet pas de rendre compte de la complexité géométrique de certains objets géologiques dans le réservoir comme des chenaux ou des méandres. C'est pourquoi, dans les quatrième et cinquième parties de ce rapport, nous abordons le problème de la simulation sous un angle différent. On considérera ainsi une statistique multipoint.

Chapitre 4

4. Synthèse de texture Mono-Echelle

Comme expliqué précédemment, les méthodes géostatistiques basées sur l'utilisation d'un variogramme ne prennent en compte que les corrélations entre deux points du réservoir. Les objets complexes de forme curvilinéaire tels que les chenaux ne peuvent être modélisés par deux points. Les méthodes de simulation multipoints, qui dépendent de plus de deux points, ont été proposées pour pallier cette difficulté. L'idée principale consiste à utiliser une image d'entrainement pour y analyser les statistiques multipoints. Puis, ces statistiques sont reproduites pendant l'étape de simulation.

Dans ce chapitre on se propose d'adapter une méthodologie multipoints, appelée synthèse de texture, pour la modélisation de réservoir. Cette approche largement utilisée dans le domaine du logiciel depuis une dizaine d'années s'est développée, en parallèle/en même temps que les nouveaux algorithmes de simulation multipoints développés en géostatistique.

La section suivante fait l'état de l'art de ces méthodes selon leur appartenance à l'infographie ou aux géostatistiques pour le réservoir. Puis on décrit en détails les étapes de l'algorithme mis au point. On procède à une analyse de sensibilité des différents paramètres, ainsi qu'à une étude de la mesure de distance entre deux motifs. Finalement on explique et on teste deux techniques pour diminuer le temps de calcul.

4.1. Etat de l'art

4.1.1. Géostatistiques multipoints

L'idée première pour modéliser les chenaux a été de modéliser des objets au lieu de simuler une valeur à un point donné. Le développement des méthodes appelées méthodes de génération par objet débute à notre connaissance avec Bridge et Leeder en 1979, et se poursuit avec Haldorsen et Damsleth (1990), Omre (1991) ou encore Deutsch et Wang (1996). Elles consistent à définir la forme de l'objet sur la base de quelques paramètres géométriques. En raison des incertitudes, les paramètres sont caractérisés par les probabilités déduites des données ou des observations. Puis ces objets aux formes tirées aléatoirement, sont positionnés sur un ensemble de points obtenus par un processus de Poisson. Bien que séduisante cette méthode se heurte à deux grandes difficultés. Le calcul des paramètres géométriques est loin d'être simple, ainsi que le conditionnement aux données dures.

Vient ensuite l'idée d'utiliser une statistique multipoint déduite à partir d'une image d'entrainement pour la simulation géostatistique. L'image d'entrainement est une représentation conceptuelle des structures spatiales attendues dans le réservoir. Elle peut provenir d'une simulation précédente, d'une image satellite, d'une image élaborée par un géologue en fonction de la connaissance qu'il a du mileu, *etc*. Cette méthode a été initialement proposée par

Guardiano et Srivastava en 1993. La motivation de ces auteurs était d'étendre la méthode de simulation séquentielle par indicatrices pour rendre compte d'objets géologiques d'architecture plus complexe sans construire complètement et explicitement un modèle de fonction aléatoire non-Gaussienne (Journel, 2005). Les premiers algorithmes de simulation multipoints impliquaient de très long temps de calcul, car il fallait scanner l'image d'entrainement à chaque fois qu'une valeur devait être simulée pour une maille. Le premier algorithme efficace, appelé SNESIM, pour Single Normal Equation Simulation, a été développé par Strebelle (2000) et permet de simuler des caractéristiques complexes pour des variables discrètes. Le point clé de cet algorithme est que les diverses configurations ou événements de faciès extraits de l'image d'entrainement sont stockés dans une structure arborescente. Cette base de données ordonnée est ensuite utilisée pour calculer les fonctions de densité de probabilité conditionnelles lors de la simulation. Cet algorithme marque une étape importante dans l'utilisation des méthodes géostatistiques multipoints, car il les rend accessibles en termes de temps de calcul, même si l'utilisation de la RAM reste un de ces désavantages majeurs. De nombreux travaux récents cherchent à améliorer cet algorithme. Straubhaar *et al.* (2011,2013), dans IMPALA, ont proposé de remplacer la classification arborescente par une classification mixte en liste et en arbre, pour pallier le problème de stockage en mémoire d'un arbre pour des images 3D.

En 2007, Arpat et Caers abandonnent le cadre probabiliste (reproduction de statistiques) au profit de la reproduction de motifs utilisée en infographie. L'image d'entrainement est utilisée comme base de données de motifs, un motif étant un morceau d'image (comme un puzzle mais de forme fixe). Une propriété importante de ce motif est qu'il doit caractériser les variations spatiales de la géologie du réservoir. Pour simuler une réalisation, chaque pixel est visité de façon aléatoire, et pour attribuer une valeur à ce pixel on colle un des motifs de l'image d'entrainement directement dessus, au lieu de tirer une valeur suivant un modèle de probabilité. Toute la qualité de l'algorithme, et de la reproduction des caractéristiques de l'image d'entrainement, tient à la façon de choisir le motif à coller et de le coller en tenant compte des données dure. Selon Arpat et Caers, la méthode est cependant plus lente que SNESIM, et la méthode ne résout pas les conflits qu'il peut y avoir entre le motif collé et les données dures. Pour diminuer le temps de simulation, les auteurs proposent d'utiliser une solution proposée par Zhang *et al.* (2006) avec FILTERSIM. Le coût de calcul est réduit grâce à un filtrage : le motif qui peut faire jusqu'à une centaine de pixels est filtré et sa représentation est alors restreinte aux scores issus des différents filtres. Le gain en temps de calcul est important mais au détriment de la qualité de reproduction de l'image d'entrainement. Dans le même courant d'idée en 2009, Gloaguen et Dimitrakopoulos utilisent la transformée en ondelettes discrètes à la place des filtres pour réduire les dimensions des motifs.

En 2010, Honarkhah et Caers proposent de jouer non pas sur la dimension du motif mais sur le nombre de motifs que l'on va comparer lors de la simulation. Ils introduisent la classification par noyaux k-means et une mesure de distance entre deux motifs, qui leur permettent de regrouper les motifs selon leur ressemblance. Lors de la simulation d'un ensemble de pixels seuls les motifs du groupe le plus proches sont utilisés. En 2010, Mariethoz *et al.* proposent dans leur algorithme d'échantillonnage direct, de ne scanner qu'une partie de l'image d'entrainement et surtout de ne pas choisir le meilleur motif mais le premier qui est en-dessous d'une distance choisie au préalable. Cette méthode permet de diminuer la mémoire occupée car l'image d'entrainement n'est pas stockée sous forme de base de données et de plus elle permet d'augmenter la variabilité des motifs reproduits en ne prenant pas le motif le plus proche mais seulement un motif assez proche.

En 2012, Tahmasebi *et al*. Proposent l'algorithme CCSIM (cross-correlation simulation) où ils n'utilisent plus la mesure de distance entre deux motifs mais une fonction de corrélation-croisée plus rapide à calculer. De plus ils utilisent un chemin régulier pour visiter les pixels et lorsqu'ils

arrivent près d'une données dures, s'il n'y a pas de motifs qui s'ajustent à la donnée, ils réduisent la taille du motif jusqu'à en trouver un qui s'y ajuste, le cas limite étant un motif réduit à un pixel.

En parallèle, des concepts similaires sont apparus dans l'infographie et traitement de l'image, regroupés sous le nom d'algorithmes de synthèse de texture. La synthèse de texture est plus généralement utilisée dans des domaines tels que les jeux vidéo, le cinéma (film d'animation) ou encore le design d'objet.

4.1.2. La synthèse de texture

Les travaux avant-gardistes de Shannon en 1948 lancent les bases de la synthèse de texture. Ils visent à reproduire un texte en utilisant la méthode des *n*-gram. L'idée est que *n* lettres consécutives (un mot) déterminent complètement la distribution des lettres suivantes. Un problème majeur est l'obtention des tables de probabilités pour chaque n-gram. Cette approche requiert beaucoup d'échantillons, probablement autant que dans un livre par exemple.

En 1974 Catmull, et plus tard en 1976, Blinn et Newell, introduisent le placage de texture (texture mapping) qui sert à habiller un objet en 3D à l'aide d'échantillons de texture 2D. La texture est appliquée sur l'objet en déformant l'image de telle sorte qu'elle épouse les surfaces courbes de l'objet. Cette méthode tient aussi compte de la direction de la lumière pour rendre à l'objet son effet 3D, ainsi que des propriétés de réflexion de la surface. Cependant, elle génère des problèmes de paramétrisation en 2D comme des distorsions de l'image pour des surfaces complexes (par exemple, pour des surfaces bicubiques ou de genre élevé). Elle ne s'applique correctement que lorsque la surface à habiller est de même aire que l'image de la texture. Si la surface de l'objet est plus grande que celle de la texture, il faut paver l'objet afin de conserver un certain niveau de détail. En effet, on comprend aisément qu'on ne peut pas trop agrandir une image sans qu'elle se pixélise. Néanmoins, cette méthode n'est pas toujours une solution puisqu'elle tend à créer des contours artificiels sauf si la texture est répétitive. Même dans ces conditions, l'œil humain perçoit très facilement les mauvais alignements et les répétitions (Palmer, 1999) ce qui réduit la qualité de l'image créée.

Une solution (Heeger et Bergen, 1995) consiste à générer une texture semblable à l'échantillon de départ directement sur la surface de l'objet. La texture générée n'est pas une copie ou un pavage. Il suffit que l'image échantillon soit suffisamment grande pour capturer le motif principal de la texture. Toutes les méthodes dérivées de cette idée relèvent de la génération de textures.

Dans sa thèse, Duranleau (2008) regroupe les méthodes de génération de texture en fonction des catégories suivantes :

- Méthodes paramétriques : basées sur un modèle statistique donné, elles génèrent de nouvelles textures selon la distribution des pixels (équivalents aux mailles d'une grille) observés en réponse à divers filtres simulant la perception humaine. Une description assez complète de ces méthodes a été rapportée par Portilla et Simoncelli (2000).

- Génération par pixel : basées sur l'échantillonnage de la texture et la réorganisation des pixels
    - méthodes multi-résolutions : construction d'une pyramide de la structure source et remplissage de la pyramide de la texture à générer niveau par niveau.

    - méthodes par recherche de voisinage : recherche un pixel dans la texture source dont le voisinage est similaire au voisinage courant du pixel à générer.
    - optimisation globale : l'échantillonnage de texture se fait par l'entremise d'une optimisation itérative sur l'ensemble de la texture générée.

- Génération par patch : même principe que la génération par pixel, mais au lieu de générer la réalisation pixel par pixel, on la génère patch par patch (*i.e.*, par groupe de pixels).

- Méthodes par analyse structurale : enrichissement des méthodes précédentes par analyse de la texture source pour en extraire la structure et guider ou modifier la génération en fonction de cette information.

- Génération adaptée aux surfaces : cette méthode améliore aussi la précédente en introduisant une déformation de l'image pour tenir compte de la courbure de la surface pour les objets 3D.

Nous nous concentrons ici sur les méthodes de génération par pixel ou par patch, qui sont les plus adaptées au contexte de modélisation de réservoir. Dans la suite nous détaillons les méthodes par recherche de voisinage. Les méthodes multi-résolution qui se prêtent bien au contexte multi-échelles seront détaillées dans le chapitre 5 section 5.1 .

Pour simuler un pixel dans les méthodes de recherche par voisinage, on regarde ses voisins et on lance la recherche dans l'image d'entrainement pour trouver un ensemble de pixels candidats dont le voisinage est similaire. La valeur d'un des pixels candidats est alors attribuée au pixel simulé. On passe ensuite au pixel suivant. Le type de voisinage, la méthode de recherche et l'ordre de génération varient selon la méthode employée.

En 1999, Efros et Leung proposent un algorithme de synthèse de texture qui donne de très bons résultats sur la réparation d'images (hole-filling). Leur voisinage est un carré centré sur le pixel en cours de simulation. Tous les pixels déjà simulés contenus dans ce voisinage servent pour la comparaison avec l'image d'entrainement. A chaque simulation de pixel l'image d'entrainement est scannée et les pixels dont le voisinage est proche de celui simulé sont extraits. A partir de ces informations, on construit une fonction de densité de probabilité pour la valeur du pixel simulé. On fait ensuite un tirage aléatoire, pour lui attribuer une valeur. Cet algorithme a un gros défaut, il est très lent et ne marche que pour les textures assez répétitives. Une méthode plus rapide a été proposée par Wei et Levoy (2000). Les pixels à simuler sont visités selon un chemin unilatéral et le voisinage à une forme en 'L', de sorte que seuls les pixels qui viennent d'être visités sont pris en compte. De plus, ils introduisent eux aussi la classification par arbre des motifs extraits de l'image d'entrainement.
De nombreux travaux portent sur l'accélération de l'algorithme de Wei et Levoy (2000). On citera Ashikmin (2001), qui propose de ne scanner que les voisinages des voisins du pixel à simuler. Zelinka et Garland (2004) construisent une carte de saut, c'est-à-dire que toutes les distances motifs à motifs sont calculées dans un premier temps, et sauvegardées. Lors de la simulation, ces auteurs vont s'en servir pour guider la recherche de motifs à scanner. Un état de l'art sur les algorithmes de synthèse de texture a été fait en 2009 par Wei *et al.*, pour plus de détails sur les autres méthodes de synthèse de texture.

En 2014, Mariethoz et Lefebvre passent en revue les liens qui existent entre la synthèse de texture et les géostatistiques multipoints. Ils soulignent le fait que les deux domaines partagent beaucoup de points communs mais ont été développé selon des axes différents. Les

algorithmes de synthèse de texture doivent être très efficaces en temps de calcul et très bons en reproduction de motifs, alors que les algorithmes de géostatistiques multipoints eux doivent intégrer les données dures et produire des réalisations stochastiques 3D.

C'est en 2007 que deux algorithmes de géostatistiques multipoints font état de leur lien avec l'infographie. Il s'agit de l'algorithmes SIMPAT de Arpat et Caers (2007) et d'un autre développés par Daly et Knudby (2007). Les différences sont dans le chemin de visite aléatoire pour l'intégration des données qui ne se fait que très peu en infographie. C'est Parra et Ortiz en 2011 qui proposent une adaptation d'un algorithme de synthèse de texture pour la simulation conditionnelle de réservoir. Ils utilisent un chemin de visite régulier comme pour Wei et Levoy (2000), mais leur voisinage est composé de deux 'L' emboités. Le premier 'L', appelé voisinage causal, contient les pixels que l'on vient de simuler. Le deuxième 'L', appelé voisinage non-causal, contient les pixels à venir. Lorsque le voisinage non-causal est vide, *ie* il ne contient aucune valeur, alors on ne compare que le voisinage causal à l'image d'entrainement. Lorsque le voisinage non-causal contient des données dures alors les deux voisinages sont utilisés pour la comparaison avec l'image d'entrainement. Cela permet d'avoir un temps de calcul raisonnable en ayant un plus petit voisinage dans la majorité des comparaisons, et de prendre en compte les données dures en amont pour éviter les problèmes de connectivité ou de continuité.

Un autre algorithme issu de l'infographie est étudié maintenant en géosciences, c'est la méthode d'image quilting pour la simulation conditionnelle (Mahmud *et al.*, 2014). Introduit en 2001 par Efros et Freeman, l'image quilting consiste à paver la grille avec des morceaux d'images. Sa particularité est que les patchs sont collés initialement plus grands que nécessaire et ensuite découpés de manière à minimiser l'erreur sur les bords et à assurer la continuité.

L'algorithme développé dans ce chapitre s'inspire de celui d'Arpat et Caers (2007). Il combine les techniques de simulation géostatistiques avec la synthèse de texture. Bien ancré dans la synthèse de texture, cet algorithme est conçu pour les applications de modélisation géologiques. Notre travail diffère de l'algorithme SIMPAT de par l'utilisation d'un chemin régulier de simulation et donc du challenge pour intégrer les données dures. De plus nous nous concentrons sur diverses techniques permettant de le rendre plus flexible et plus rapide, et proposons quelques modifications pour permettre l'intégration des données dures avec un chemin régulier sans perdre la continuité dans l'image finale.
Dans la première section, on explique pas à pas l'algorithme pour la simulation non-conditionnelle, puis pour la simulation conditionnelle. Dans la deuxième section, nous examinons deux méthodes pour diminuer le temps de calcul et nous les testons pour voir leur effet sur la qualité de l'image finale.

4.2. Algorithme

4.2.1. Simulation non-conditionnelle

L'image d'entrainement doit être premièrement scannée à l'aide d'un template pour en extraire des motifs et créer ainsi une base de données de motifs. Notre choix par défaut pour le template s'est porté sur une fenêtre carrée centrée sur le pixel en train d'être simulé (Figure 24).

Comme il n'y a pas de données dures, la phase d'initialisation de la grille à simuler débute par le choix d'un patch aléatoirement dans l'image d'entrainement et par son collage au milieu de la grille. Ensuite la grille est remplie couronne par couronne autour du patch collé (Figure 24). Une couronne est définie à chaque étape à partir des pixels non simulés voisins directs de pixels déjà simulés. La première couronne de pixel est simulée. Ensuite, on prend les pixels non simulés à coté et on génère de nouvelles valeurs pour ces pixels. Ce processus est répété jusqu'à ce que la grille soit entièrement simulée. Cet ordre de visite est appelé « régulier » par contraste avec l'ordre de visite « aléatoire » qui est généralement utilisé pour les simulations géostatistiques. Pour chaque pixel dans une couronne, on compte le nombre de pixels voisins déjà simulés. Puis on visite les pixels de la couronne de celui qui a le plus de voisins à celui qui en a le moins. Avant d'aller plus loin, il serait utile de mentionner le fait que la simulation se fait patch par patch au lieu de pixel par pixel, et que la taille du patch est plus petite que la taille du template. Cela permet de réduire le temps de calcul et de mieux préserver la continuité spatiale des structures géologiques dans la nouvelle image.
I est l'image simulée à partir de l'image d'entrainement *TI*. *p* est le patch de pixels en train d'être simulés sur *I* à l'itération courante, il est centré sur le pixel *i*. Le cas limite apparaît lorsque *p* ne contient que *i*. *w* désigne le template. Lorsqu'il est centré sur *i*, on appelle *w(i)* les pixels voisins de *i* appartenant à *w*. Les étapes successives de l'algorithme sont listées ci-dessous.

Algorithme :

- Initialisation: sélectionner un patch aléatoirement dans la *TI* et le coller au milieu de la grille à simuler *I*. Le patch pourrait être collé n'importe où ailleurs sur la grille.
- Jusqu'à ce que tous les pixels soient visités une fois
    - A partir des pixels déjà simulés, identifier une couronne de pixels voisins non simulés
    - Trier ces pixels selon leur nombre de voisins déjà simulés, de celui qui en a le plus à celui qui en a le moins
    - Les simuler un à un selon cet ordre
        - Identifier les voisins *w(i)* du pixel courant selon le template choisi
        - Comparer *w(i)* à la base de données des motifs extraits de la *TI* et sélectionner le plus proche de *w(i)*. La distance *d* entre les motifs et *w(i)* est définie par une mesure donnée. Ici par défaut le choix est la norme L2 où chaque terme a un poids qui dépend de son éloignement au pixel simulé. Ce point sera développé dans la section 4.3.2
        - Sur le motif choisi, le patch central est extrait et collé sur le groupe de pixel autour de *i* sur *I*. Précision : les pixels déjà simulés ne sont pas effacés, c'est-à-dire que l'on ne colle de nouvelles valeurs que sur les pixels qui n'en contiennent pas.

Jusqu'à maintenant toute l'image d'entrainement devait être scannée pour chaque patch simulé pour identifier le motif le plus proche du template. D'autres stratégies sont envisagées pour améliorer le temps de calcul et exposées dans la section 4.4.

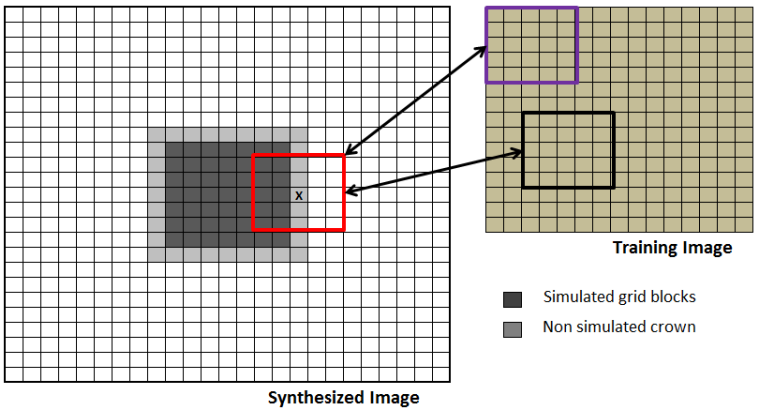


Figure 24 Schéma de synthèse pour attribuer une valeur à un patch. Sur la gauche, l'image synthétisée *I*. Sur la droite l'image d'entrainement *TI*. On veut attribuer une valeur au patch *p* (*i* est marqué par un x) en comparant *w(i)* (ici en rouge à gauche) aux motifs extraits (ici en exemple violet et noir) de l'image d'entrainement (à droite).

A ce stade on peut préciser que chaque fois que l'on change le germe de la fonction de génération de nombres pseudo-aléatoires, on change à l'initialisation le motif extrait de l'image d'entrainement. Cela se traduit par la génération de différentes réalisations. Cependant, une fois le motif initial choisi, le processus de simulation défini ci-dessus est déterministe : en effet lorsque l'on remplit la grille on choisit les meilleurs patchs, ceux qui ont la plus petite distance au voisinage. Une variante de l'algorithme qui permet d'utiliser les divers algorithmes de perturbations tels que la déformation graduelle (Hu, 2000) ou la méthode de perturbation des probabilités (Caers, 2003), est présenté section 4.2.3.

4.2.2. Simulation conditionnelle

L'utilisation d'un chemin de visite aléatoire est habituellement recommandée pour permettre l'intégration des données dures. Dans cette section, nous montrons comment prendre en compte les données dures tout en gardant un chemin régulier et un processus de génération par couronnes.

Intuitivement, on voit que la taille du template est directement liée au temps de calcul. Par conséquent, on va préférer de petits templates pour réduire le temps de simulation. Cela veut dire d'un autre coté que si on commence l'initialisation au centre de la grille et que l'on grandit par couronnes concentriques, les données dures ne seront détectées que lorsqu'elles seront proches du pixel en cours de simulation. Donc à un moment donné autour du pixel simulé, on aura tous les pixels qui ont déjà été simulés sans tenir compte de la donnée dure qu'ils n'ont pas vus, et la donnée dure qui peut être très différente. Il est fort à parier que cela va créer de grosses discontinuités dans l'image.

C'est la raison pour laquelle les algorithmes de géostatistiques multipoints utilisent généralement un chemin aléatoire qui atténue cet effet. Toutefois, avec un chemin aléatoire, il faut de plus grands templates pour capturer correctement les structures de l'image d'entrainement, ce qui signifie un temps de calcul plus important. Une autre option est proposée par Parra et Ortiz (2011) comme on l'a évoqué dans l'état de l'art (section 4.1.2). Ils ont proposé de considérer des voisinages carrées divisés en deux domaines : l'un contenant les pixels déjà simulés, et l'autre ceux à venir. Seul le premier est utilisé si le deuxième ne contient pas de données. Cela réduit partiellement les discontinuités. Cependant, il y a de nombreux cas où il faut réconcilier dans un même template les données dures et les pixels qui ont été simulés sans

tenir compte des données. De tels motifs ont peu de chance d'exister dans la base de données, créant ainsi de nouveau des discontinuités dans la réalisation finale.

Nous proposons une approche différente. L'étape d'initialisation change. Elle doit maintenant tenir compte de la présence de données dures. On va donc créer une figure géométrique fermée en reliant les données deux à deux par des segments. Ensuite, nous simulons des valeurs uniquement pour les pixels traversés par ces segments. Ceci est effectué en utilisant le même procédé que pour les couronnes, à savoir les pixels avec les plus de voisins non-vides sont simulés en premier. Une fois les pixels des segments simulés, on obtient une figure fermée déjà simulée et on revient au processus de construction par couronnes présenté dans la section 4.2.1. Dans ce cas, les couronnes comprennent les pixels à l'intérieur et à l'extérieur de la figure fermée.

Un exemple avec trois données dures est représenté sur la Figure 25. Les pixels gris sont les pixels non simulés, les noirs représentent les chenaux et les blancs le milieu encaissant. Dans la Figure 25 a), on voit la grille non simulée et l'emplacement de 3 données dures qui signalent l'emplacement de chenaux. Dans la Figure 25 b) on voit le résultat de l'étape de création de la figure fermée et de la simulation des pixels traversés par chaque segment de la figure. Dans la Figure 25 c) et d) on n'a illustré que la simulation des couronnes intérieures pour plus de clarté. Dans la Figure 25 e) on voit la simulation de la première couronne extérieure. Enfin dans la Figure 25 f) on voit la réalisation finale, les données dures sont signalées par des points rouges. On voit que les données sont bien respectées.

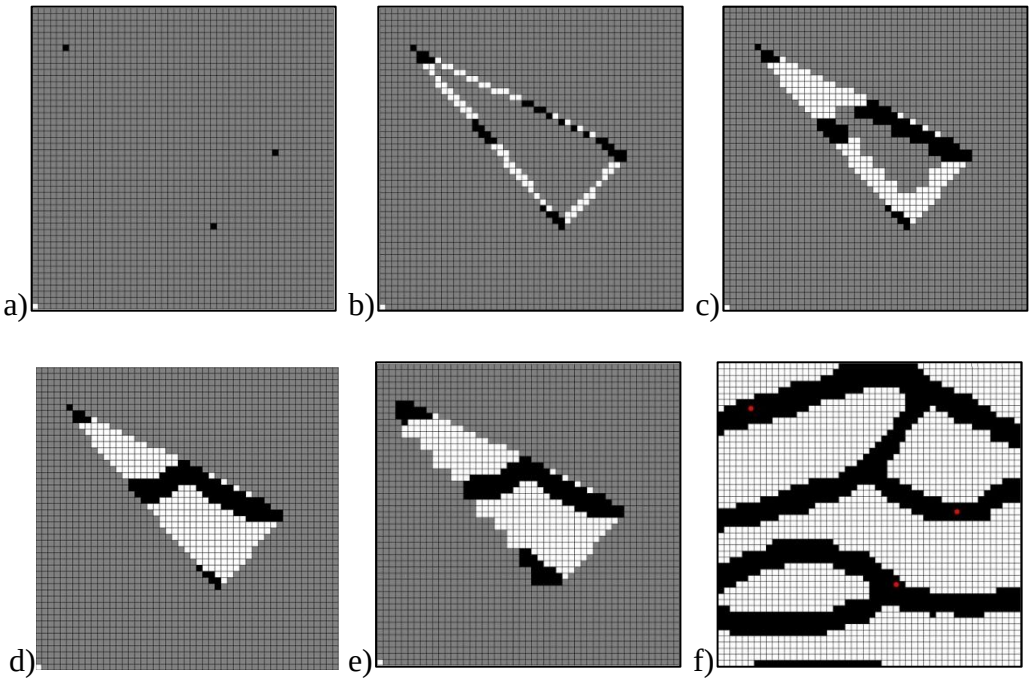


Figure 25 Schéma illustrant les étapes successives suivies pour la simulation conditionnelle. a) Emplacement des 3 données dures sur une grille de 50 × 50. b) Construction du triangle et simulation des pixels le long des segments avec un template de taille 19 × 19 pixels. c à d) Simulation de l'intérieur du triangle en propageant par couronnes. e) Simulation de la première couronne extérieure f) La réalisation finale (points rouges : emplacements de données).

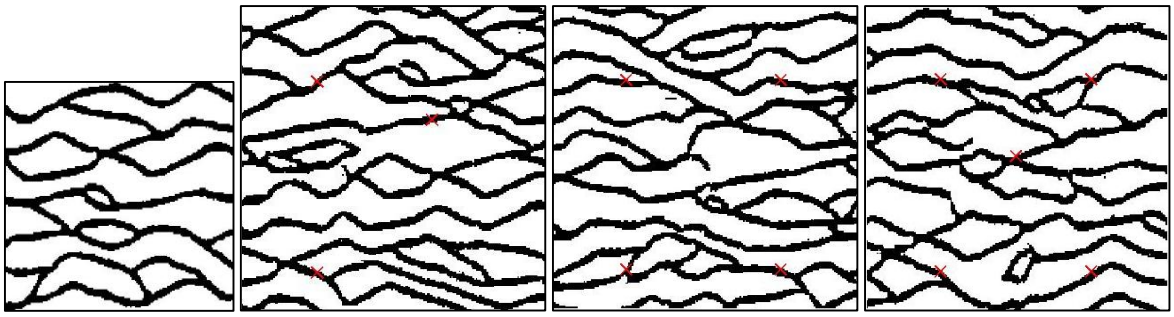


Figure 26 Simulation Conditionnelle. Première image à gauche : image d'entrainement. Autres images : réalisations simulées avec contraintes sur respectivement 3, 4 ou 5 données dures. Les croix rouges représentant les emplacements des données. Taille du template : 19×19 pixels; Taille du patch : 5×5 pixels.

D'autres exemples de réalisations conditionnelles sont présentés dans la Figure 26 avec trois, quatre et cinq données dures indiquant la présence de chenaux. Les réalisations finales obtenues avec l'algorithme proposé sont cohérentes avec les structures représentées dans l'image d'entrainement et respectent les données.

4.2.3. Déformation Graduelle

Comme dit précédemment l'algorithme présenté ci-dessus est déterministe car on choisit à chaque simulation de patch, le motif le plus proche du voisinage du patch. Pour rendre cet algorithme compatible avec la méthode de déformation graduelle, on modifie l'étape de sélection du meilleur motif.

Lors de cette étape de sélection, au lieu de garder la meilleure adéquation entre le voisinage et les motifs extraits de l'image d'entrainement, nous gardons les *n* plus proches motifs et en choisissons un au hasard parmi ceux-ci. Le choix n'est pas totalement aléatoire. On calcule un poids pour chacun des *n* motifs extraits qui dépend de la distance entre le motif et le voisinage du patch simulé. Plus le motif est loin du voisinage plus son poids est faible, moins il a de chance d'être sélectionné.

Pour illustrer l'effet du choix parmi les 10 meilleurs motifs, sur les réalisations, nous prenons quatre données dures, signalant quatre emplacements pour les chenaux, et simulons quatre réalisations avec quatre jeux de nombres aléatoires différents. Les résultats sont montrés Figure 27. On voit que l'on obtient quatre réalisations différentes, qui respectent les données et qui restent semblables à l'image d'entrainement. On observe quelques discontinuités de chenaux, dues au fait que l'on ne prend pas le motif le plus proche du voisinage.

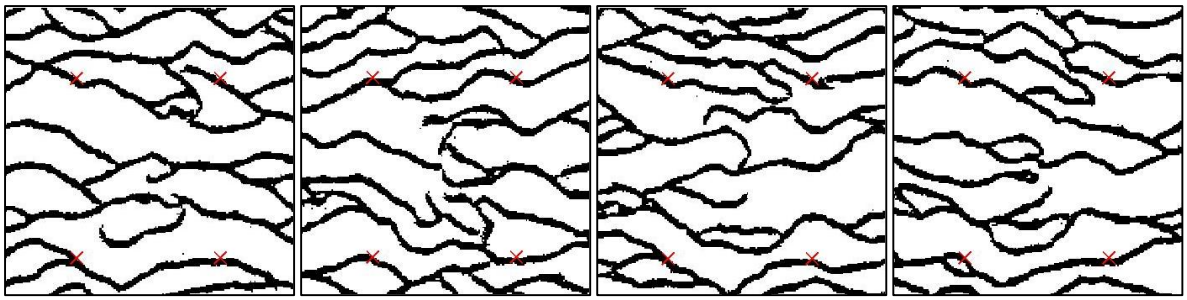


Figure 27 Quatre réalisations conditionnées sur quatre données dures en prenant quatre bruits blancs Gaussiens différents. Template : 19x19. Taille de Patch: 5x5

On rappelle brièvement que la méthode de déformation graduelle est une technique utilisée pour perturber des réalisations en géostatistique avec un nombre réduit de paramètres. Elle est basée sur la combinaison linéaire de bruits blancs Gaussiens indépendants (rappel de l'Equation 8) :

$$z(\theta) = z_1 \cdot cos(\pi\theta) + z_2 \cdot sin(\pi\theta)$$

où z_1 et z_2 sont des bruits blancs Gaussien indépendants et θ est le paramètre de déformation généralement compris entre -1 et 1.
Le fait d'utiliser un jeu de nombres aléatoires tirés d'une distribution uniforme pour choisir parmi *n* motifs, n'implique que peu de modifications pour utiliser la déformation graduelle.

Introduisons quelques notations: *G* est la fonction de répartition de la loi normale standard et u_i un jeu d'éléments tiré de la loi uniforme.

$$z(\theta) = G^{-1}(u_1) \cdot cos(\pi\theta) + G^{-1}(u_2) \cdot sin(\pi\theta)$$
$$u(\theta) = G\big(z(\theta)\big) \qquad \textbf{Equation 18}$$

On transforme donc les u_i en bruits blancs Gaussien à l'aide de la fonction probit G^{-1}. On fait une déformation graduelle pour trouver le θ optimal et on peut revenir à un jeu d'éléments uniformes en utilisant la fonction de répartition G.

4.3. Analyses de sensibilité des paramètres du modèle

Dans cette section nous discutons des performances de l'algorithme décrit ci-dessus, et de sa sensibilité sur certains paramètres comme la taille du template ou encore la mesure de la distance *d* entre les motifs et le template *w(i)*. Nous montrons l'influence du chemin de visite pour la simulation des patchs sur la qualité de l'image obtenue.
Cette analyse est basée sur un ensemble de simulations effectuées à partir de la même image d'entrainement. Elle est connue et utilisée pour illustrer le potentiel des méthodes de géostatistiques multipoints (Figure 28). Elle représente des chenaux de haute perméabilité contenu dans une matrice très peu poreuse/perméable. L'image d'entrainement fait 150×160 pixels. Dans tous les cas considérés, la réalisation fait 200×200 pixels.

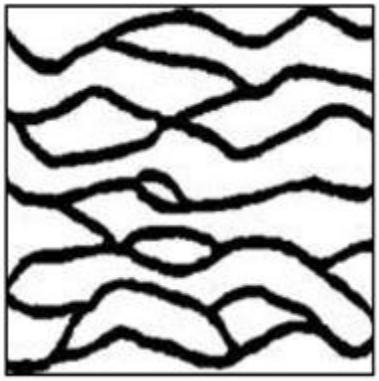


Figure 28 Image d'entrainement représentant des chenaux dans un milieu argileux

4.3.1. Quelle est l'influence de la taille du template et de la taille des patchs ?

L'algorithme présenté dans la section 4.2 implique la définition de paramètres tels que la taille du template ou encore la taille du patch collé à chaque simulation.
Les résultats de la Figure 29 montrent que la taille du voisinage est importante. Clairement un voisinage trop petit ne permet pas de capturer les hétérogénéités de grande taille présentes sur l'image d'entrainement. L'image synthétisée avec un template 5×5 (Figure 29) ne présente aucun chenal. Augmenter la taille du template de 9×9 à 19×19 (Figure 29 b et c) permet de faire apparaitre des chenaux et d'améliorer la continuité et la connectivité de la géométrie des chenaux. Cependant augmenter la taille du template diminue le nombre de motifs extraits de l'image d'entrainement. Par exemple considérons une image d'entrainement de taille 150×160, et une taille de template de 9×9, il en résulte 21 584 motifs extraits. Maintenant pour cette même image d'entrainement, augmentons la taille du template à 31×31, le nombre de motifs extraits est alors de 15 600, soit 30% de moins. On a alors un problème de représentativité. Si la base de données des motifs extraits ne contient pas assez de motifs, il est plus difficile de trouver le motif approprié lors de la simulation d'un patch. Il en résulte des discontinuités dans l'image finale (réalisation) comme dans la Figure 29 e). Une taille de template appropriée est un compromis entre les deux. Elle doit être suffisamment grande pour capturer l'hétérogénéité mais suffisamment petite pour qu'il y ait un nombre suffisant de motifs extraits.

Ici nous avons procédé par un test exhaustif de toutes les tailles de templates entre 5× 5 et 31×31, pour trouver la taille de fenêtre optimale. Il s'avère qu'un template de 19×19 donne les meilleurs résultats. Cependant quelques auteurs ont développé des méthodes pour trouver cette taille de template (Tan *et al.*, 2013). On voit aussi dans la Table 1 le temps de calcul augmente avec la taille du template.

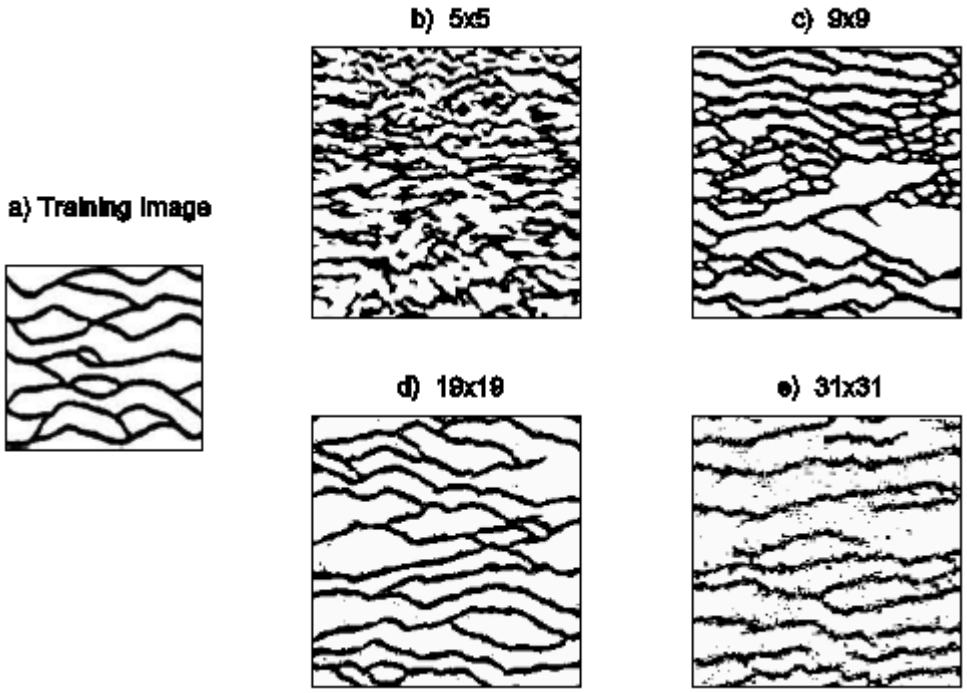


Figure 29 Influence de la taille du template. A gauche : image d'entrainement (a). Autres images : réalisations simulées avec une taille de template grandissante de b) 5×5 à e) 31×31. Taille de patch fixe : 5×5. Chemin de visite régulier.

Table 1 Impact de la taille du template sur le temps de calcul. Le temps de référence T_0 est relatif au temps de simulation avec le plus petit template 5×5 avec un patch 3×3

Taille du template	5×5	9×9	19×19	31×31
CPU-time	T_0 =42sec	$3 \cdot T_0$	$9 \cdot T_0$	$15 \cdot T_0$

On a estimé la complexité de l'algorithme à partir de deux tests. Dans le premier test, illustré Figure 30 à gauche, on trace le temps de simulation en fonction du nombre de pixels contenus dans le template. On voit clairement que la courbe n'est pas linéaire mais plutôt logarithmique. Or on sait que la complexité des algorithmes de synthèse de texture usuels est directement proportionnelle aux nombres de pixels dans le template. Dans un deuxième test, illustré Figure 30 à droite, on trace le temps de calcul en fonction du nombre de pixels dans le template mais à nombre de motifs dans la base de données fixe. On a montré ci-dessus que lorsque l'on augmente la taille du template on diminue le nombre de motifs extraits dans la base de données. Donc on peut expliquer la décélération du temps de calcul dans le test 1 pour les grands templates par le fait qu'il y a moins de templates à comparer même si ceux-ci contiennent plus de pixels. C'est pourquoi dans le test 2, on fixe le nombre de motifs dans la base de données, de façon artificielle, au nombre de motifs qu'il y a dans la base de données avec le plus grand template utilisé (*ie* avec le template de 31×31, on a 15 600 motifs) même lorsque l'on simule avec de plus petits templates. On voit clairement que le temps de simulation augmente linéairement avec le nombre de pixels dans le template (Figure 30 à droite).

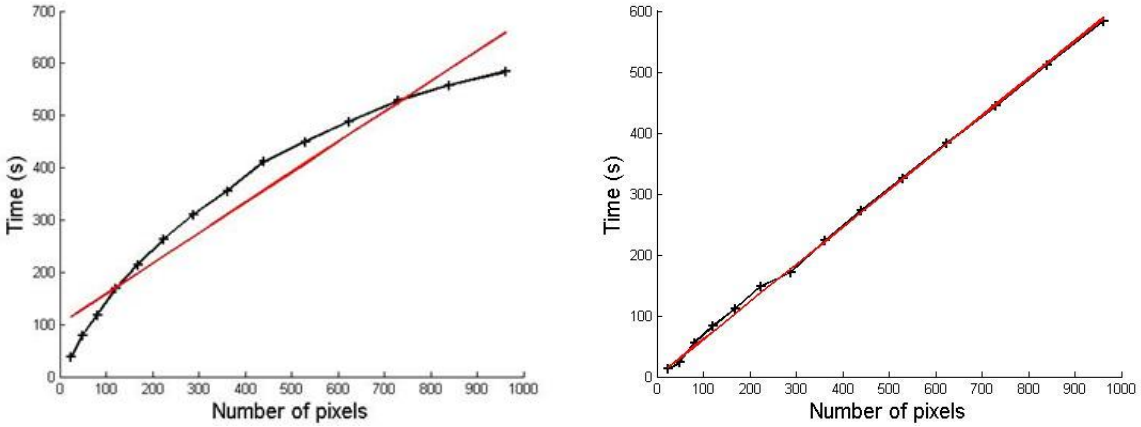


Figure 30 Complexité de l'algorithme. On trace le temps de simulation en fonction du nombre de pixels dans le template. Dans le test à gauche le nombre de motifs dans la base de données décroit. Dans le test à droite on fixe artificiellement le nombre de motifs dans la base de données. En noir les résultats numériques, en rouge la droite de régression.

La taille des patchs collés est aussi un paramètre qui a son importance. Les cas les plus extrêmes sont lorsque le patch est réduit à un pixel ou alors au contraire lorsque le patch fait la même taille que le template.
Quand le patch est trop petit, le temps de simulation est très long. Quand le patch est trop gros, l'image créée devient une copie conforme de l'image d'entrainement, on perd la notion de procédé aléatoire : la construction de l'image revient à assembler les pièces d'un puzzle et l'apparition de nouveaux motifs devient improbable. Encore une fois, la taille du patch est un compromis.

La Figure 31 montre l'influence de la taille du patch, lorsqu'on garde la taille du template fixe (ici 19×19) sur la réalisation. Lorsque l'on augmente la taille du patch de 1×1 à 5×5, la qualité du rendu final de l'image est améliorée et préservée. Dans un même temps le temps de calcul est divisé par 3 (Table 2). Les temps de calcul relatifs en fonction de la taille du patch sont reportés dans la Table 2. Pour une taille de patch de 19×19 (Figure 31 e), la réalisation ressemble à un arrangement de morceaux provenant de l'image d'entrainement sans continuité aux interfaces.

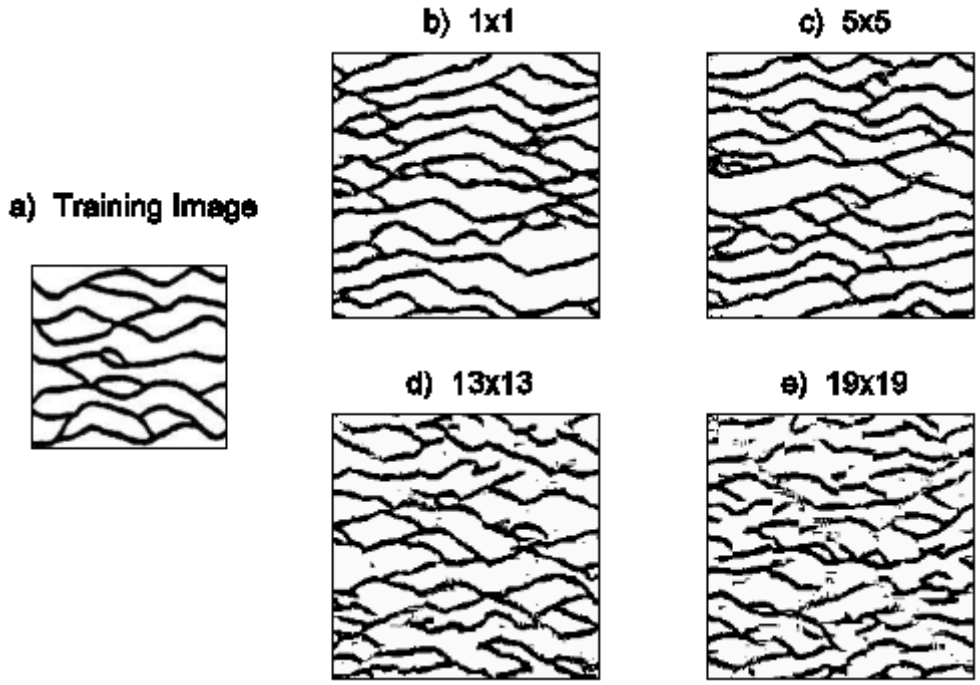


Figure 31 Influence de la taille du patch. A gauche : image d'entrainement (a). Autres images : réalisations simulées avec une taille de patch grandissante de b) 1×1, à e) 19×19. Taille du template fixe : 19×19. Chemin de visite régulier.

Table 2 Impact de la taille du patch sur le temps de calcul. Le temps de référence est celui de la simulation avec un template de taille 19×19 et un patch de taille 1×1

Taille du patch	1×1	5×5	7×7	11×11	13×13	19×19
CPU-time	T_1 =765sec	$T_1/3$	$T_1/4$	$T_1/6$	$T_1/7$	$T_1/10$

4.3.2. Comment gérer le voisinage?

Dans l'algorithme introduit dans la section 4.2, il faut comparer le template aux motifs extraits de l'image d'entrainement pour trouver le plus proche. Cette comparaison est établie sur une mesure de distance.
Plusieurs méthodes sont utilisées pour estimer cette distance, certaines étant plus appropriée que d'autres. Pour simplifier, on considère un template de 5×5 pixels et on calcule la distance entre un voisinage donné V et un motif donné T, qui contiennent tous les deux 5×5 pixels (Figure 32) .

Voisinage V

		V_3	V_4	
			V_9	V_{10}
		V_{18}	V_{19}	V_{20}
			V_{24}	V_{25}

Motif T

T_1	T_2	T_3	T_4	T_5
T_6	T_7	T_8	T_9	T_{10}
T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
T_{16}	T_{17}	T_{18}	T_{19}	T_{20}
T_{21}	T_{22}	T_{23}	T_{24}	T_{25}

Figure 32 Voisinage V et motif extrait T, utilisant un template 5×5 pixels

Tous les pixels de T ont une valeur extraite de l'image d'entrainement. Ceux de V (sur la grille simulée) peuvent ne pas contenir de valeur, s'ils n'ont pas été déjà simulés ou s'ils ne contiennent pas de données.

La distance entre V et T est donnée par la somme des différences pondérées au carré entre chaque pixel contenant une information de V et ceux de T :

$$d = \sum_{i \in N} \omega_i (v_i - t_i)^2 \qquad \textbf{Equation 19}$$

avec n le nombre de pixels dans le template, ω_i un poids, v_i la valeur du pixel i dans le voisinage V, et t_i la valeur du pixel i dans le motif T. Pour calculer les poids, on commence par leur attribuer la valeur 0 si les pixels de V associés sont vides et 1 sinon (Figure 33 a). Puis on calcule un noyau Gaussien (Figure 33 b), et enfin on multiplie les deux jeux de poids et on les normalise (Figure 33 c). Dans ce cas-ci, les pixels les plus au centre du voisinage ont plus de poids que ceux qui sont aux bords.

a) Poids uniformes

0	0	1	1	0
0	0	0	1	1
0	0	0	0	0
0	0	1	1	1
0	0	0	1	1

b) Noyau Gaussien

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

c) Poids Gaussiens

$\frac{1}{82}$

0	0	7	4	0
0	0	0	16	4
0	0	0	0	0
0	0	26	16	4
0	0	0	4	1

Figure 33 Calcul des poids du voisinage V de la Figure 6. a) Poids Uniformes. b) Noyau Gaussien de variance 1. c) Poids Gaussiens calculés à partir du noyau Gaussien

Un exemple de l'impact des poids sur le rendu de l'image finale est présenté en Figure 34. La première réalisation (Figure 34 b) est générée à l'aide de poids uniformes alors que la deuxième (Figure 34 c) est générée avec les poids Gaussiens. On peut voir qu'attribuer des poids plus forts aux pixels du centre du voisinage améliore la qualité de l'image finale.

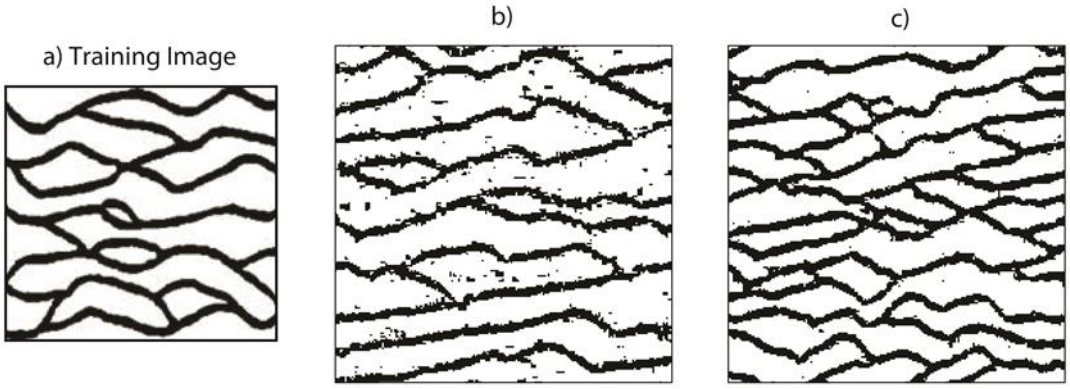


Figure 34 Impact des poids sur le rendu de l'image finale. a) L'image d'entrainement. b) La réalisation générée à l'aide des poids uniformes. c) La réalisation générée à l'aide des poids Gaussiens. Le template utilisé dans les deux réalisations est de 23×23 pixels et la taille du patch est de 5×5. L'ordre de visite est régulier.

4.3.3. Quel est l'impact du chemin de visite ?

Le procédé de simulation décrit ici est basé sur la construction successive de couronnes, ces couronnes étant définies comme un ensemble de pixels avec au moins un pixel voisin non vide. Les pixels d'une couronne sont simulés avant de déterminer la couronne suivante et ce jusqu'à ce que tous les pixels de l'image finale aient été visités une fois. Par ailleurs, le chemin de visite habituel en simulation géostatistique multipoints est aléatoire. Dans ce paragraphe, nous testons l'impact de l'ordre de visite lors de la simulation.

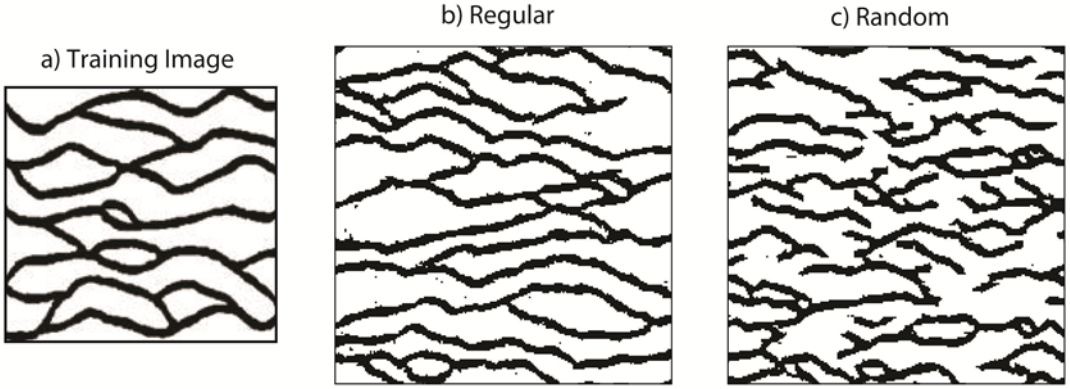


Figure 35 Impact de l'ordre de visite. a) L'image d'entrainement. b) Réalisation générée par un chemin régulier. c) Réalisation générée par un chemin aléatoire. Template : 19×19. Taille patch : 5×5.

Les deux réalisations sont montrées Figure 35, la première est obtenue avec le procédé de couronnes successives (Figure 35 b), la deuxième en visitant aléatoirement les pixels lors de la simulation (Figure 35 c). Dans les deux cas on a utilisé un template de 19×19 pixels. De façon très claire, le procédé utilisant un chemin régulier permet d'obtenir de meilleurs résultats. Il préserve la connectivité latérale des chenaux, alors que pour un chemin aléatoire les chenaux sont déconnectés. Une possibilité pour améliorer les résultats dans le cas d'un chemin aléatoire est d'augmenter la taille du template, tout en sachant que cela va augmenter les temps de calcul et la place mémoire. Dans la Figure 36 on se concentre sur le procédé avec chemin aléatoire et on augmente progressivement la taille du template. Comme on peut le constater le rendu de l'image finale s'améliore mais pas jusqu'à atteindre celui obtenu par un procédé avec chemin régulier.

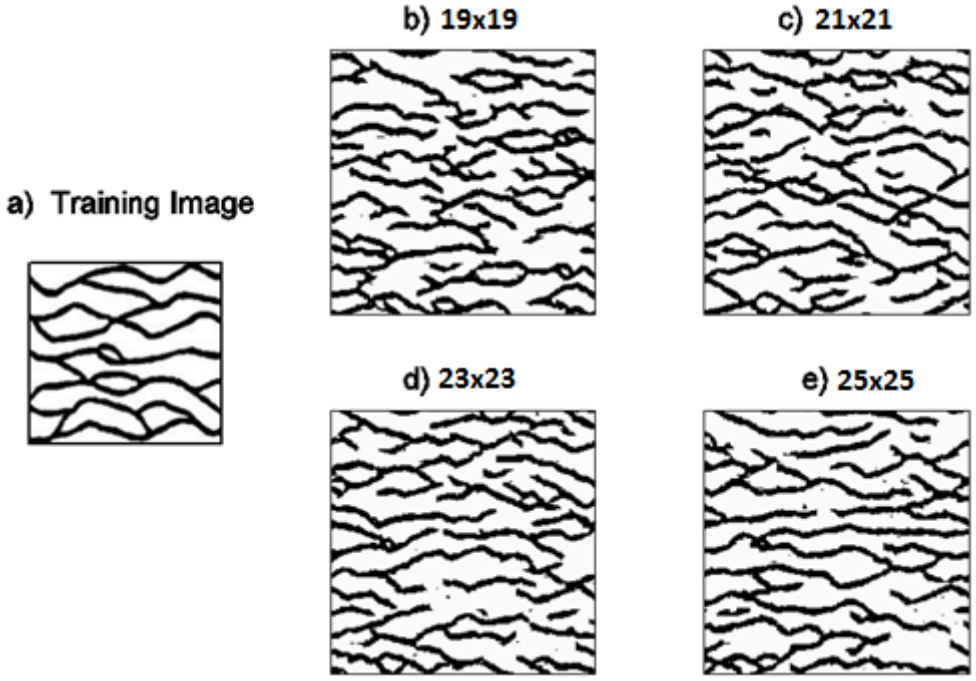


Figure 36 Influence de la taille du template sur un procédé avec chemin aléatoire. a) Image d'entrainement. b) Template 19×19. c) Template : 21×21. d) Template : 23×23. e) Template 25×25. Taille patch : 5×5. Ordre de visite : aléatoire.

L'utilisation du chemin de visite aléatoire est généralement recommandée pour permettre l'intégration des données dures. Cependant nous avons montré dans le paragraphe 4.2.2 la simulation conditionnelle reste possible pour un procédé avec couronne.

4.4. Techniques de diminution du temps de calcul

Dans cet algorithme, les temps de calcul sont très significatifs. En effet, il faut scanner toute l'image d'entrainement pour attribuer une valeur à un seul patch. L'idée générale pour diminuer le temps de calcul est de réduire le nombre de voisinages à scanner pour attribuer une valeur à un pixel. Deux solutions assez intuitives ont été testées ici. La première consiste à ne scanner qu'une partie de l'image au lieu de l'image entière (Mariethoz *et al.*, 2010). La deuxième est de classer les motifs extraits de l'image d'entrainement, soit à l'aide d'un arbre (Strebelle, 2000 ; Wei et Levoy, 2000) soit en classes (Zhang *et al.*, 2006 ; Honarkhah et Caers, 2010).

4.4.1. K-means clustering

Pour réduire le nombre de voisinages à scanner, nous suggérons de définir des classes à partir des motifs extraits de l'image d'entrainement, par exemple en utilisant l'algorithme k-means (Duda *et al.*, 2001). L'algorithme K-means est un algorithme itératif utilisé pour partitionner des objets en k classes fixées *a priori*. En quelques mots, l'algorithme sélectionne aléatoirement k points, un pour chaque groupe. Ces points sont considérés comme les barycentres initiaux des clusters. Par la suite, chaque objet est affecté au barycentre dont il est le plus proche. Cela implique la définition d'une distance. Enfin, les barycentres sont mis à jour en fonction des objets assignés à leur classe. Le processus d'attribution des objets aux barycentres et de modification des coordonnées des barycentres est répété jusqu'à convergence. Dans le cas général, les barycentres sont définis à partir de la moyenne des objets assignés à leur classe.

L'algorithme de simulation multipoints FILTERSIM (Wu *et al.*, 2008) fait également référence à l'algorithme k-means. Il se déroule comme suit. Tout d'abord, les filtres sont appliqués à l'image d'entrainement lorsqu'elle représente une variable continue ou aux cartes d'indicateurs correspondants lorsqu'elle représente une variable discrète (les cartes d'indicateurs indiquent l'absence/présence d'un faciès donné à n'importe quel endroit). Il en résulte un ensemble de scores, qui sont ensuite classés par k-means. Contrairement à Zhang *et al.* (2006) ou Wu *et al.* (2008), nous n'effectuons aucun filtrage, et la classification des motifs extraits de l'image d'entrainement n'est calculée qu'une fois. Deuxièmement, dans le cas de variables discrètes, nous n'utilisons pas de cartes d'indicateurs : les modèles de référence sont extraits de l'image d'entrainement tels quels. Enfin, pour une classe donnée, nous ne considérons pas comme représentant de la classe, le motif résultant de la moyenne point à point des motifs de la classe, mais le motif le plus proche du barycentre. Une autre différence est soulignée ci-dessous.

Nous regroupons les motifs extraits de l'image d'entrainement en k classes, chacune d'elles étant représentée par le motif le plus proche de la moyenne point à point des motifs dans la classe. Au cours du processus de simulation, chaque fois que nous affectons une valeur à un pixel donné, il faut examiner la base de données des motifs pour identifier le plus proche du voisinage du pixel cible. Si la base de données comprend 100.000 modèles, 100.000 comparaisons de distance sont nécessaires pour simuler une valeur unique, ce qui est très demandeur en temps de calcul. Lorsqu'au préalable la base de données a été classée en k classes, il ne faut que k comparaisons de distance pour trouver la classe la plus proche. Ensuite, la recherche du meilleur motif est limitée à cette classe, conduisant ainsi à une accélération significative du temps de calcul.

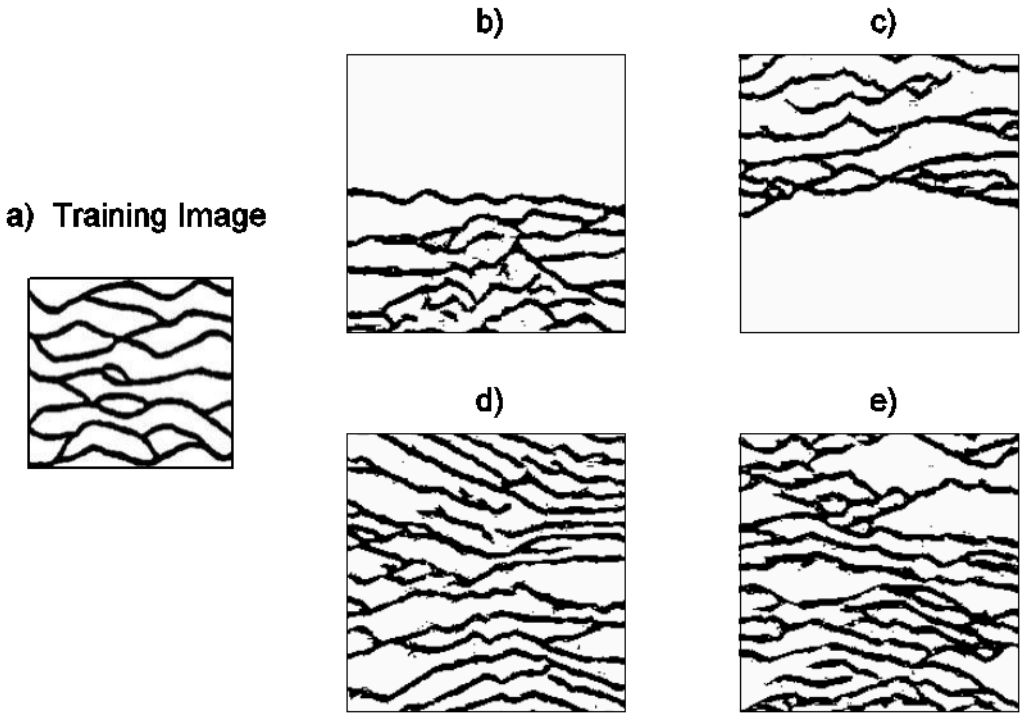


Figure 37 Simulations avec k-means clustering. a) Image d'entrainement. b) Classification avec 5 classes. c) Avec 10 classes. d) Avec 20 classes. e) Avec 30 classes. Taille du template 19×19. Taille du patch : 5×5. Chemin de visite régulier

Table 3 Temps de calcul selon le nombre de classes. Le temps de référence T_3 est donné par la réalisation générée avec 1 classe (*ie.* sans classification). Taille de template 19×19. Taille de patch : 5×5. Chemin de visite régulier.

Nombre de classes	1	5	10	20	30
Temps de calcul pour la classification et pour la simulation	0 – T_3 (255s)	0.08 T_3 - 0.86 T_3	0.18 T_3 - 0.75 T_3	0.33 T_3 - 0.33 T_3	0.66 T_3 - 0.20 T_3

Une première série de tests avec un nombre croissant de classes est représentée sur la Figure 37. Les temps relatifs de calcul sont rapportés dans le Table 3. Comme on s'y attendait, la classification contribue à rendre le processus de simulation plus rapide, même si elle nécessite une phase de classification préliminaire. Cela est d'autant plus avantageux qu'il n'est pas nécessaire de répéter cette phase préliminaire lors de la génération de plusieurs réalisations. La Figure 37 montre aussi une caractéristique particulière. Les réalisations obtenues avec un petit nombre de classes (5 et 10) montrent des chenaux, mais seulement dans une partie de la grille de simulation. La qualité des réalisations s'améliore avec le nombre de classes. Une explication possible est que le tri des motifs est moins discriminant quand il y a quelques classes seulement. Ainsi, des motifs similaires peuvent être répartis dans des classes distinctes. Cependant, même si les résultats sont meilleurs quand il y a plus de classes, la continuité latérale de chenaux n'est pas parfaitement reproduite.

En nous référant aux travaux de Ashikhmin (2001) nous introduisons le concept de cohérence. Le principe de base est très simple. Les valeurs affectées à des pixels voisins dans la réalisation ont tendance à être extraites de la même région dans l'image de la formation. Par conséquent, Ashikhmin (2001) a réduit la recherche de la meilleure configuration à une liste très courte contenant seulement les motifs associés aux voisins. De même, nous suggérons l'extension de la recherche du meilleur motif aux classes desquelles les voisins ont été extraits. En fait, nous pouvons restreindre notre attention sur les classes correspondant aux voisins les plus proches. Dans ces conditions, le choix du meilleur motif implique la recherche d'un maximum de quatre classes. Ceci dégrade légèrement les performances de l'algorithme, mais le temps de calcul reste raisonnable. Pour en revenir à l'exemple décrit ci-dessus avec 30 classes, le temps de calcul nécessaire pour simuler une réalisation augmente de 0,20 T_3 à 0,37 T_3. Ceci contribue cependant à améliorer de manière significative les caractéristiques des réalisations générées (Figure 38) : la continuité latérale et la connectivité des chenaux obtenus sont beaucoup plus cohérentes avec ceux de l'image d'entrainement.

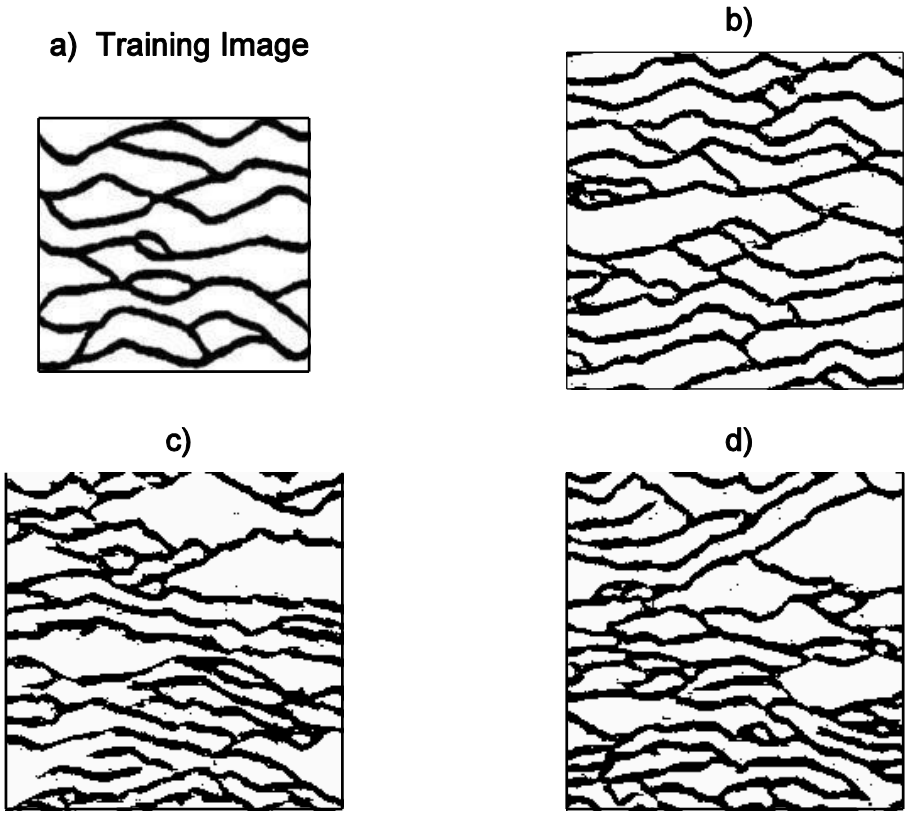


Figure 38 Impact de la cohérence et de la classification. a) Image d'entrainement. b) Simulation sans classification. C) Classification sans cohérence : 30 classes. D) Classification et cohérence : 30 classes. Taille du template : 19×19. Taille du patch : 5×5. Chemin de visite régulier.

4.4.2. Scan partiel de l'image d'entrainement

L'autre possibilité proposée pour réduire le temps de calcul consiste à ne balayer qu'une partie de l'image d'entrainement lors de la recherche du meilleur motif (Mariethoz *et al.*, 2010). Dans ce cas, il n'est pas nécessaire de créer une base de données pour stocker les motifs provenant de l'image d'apprentissage. Chaque fois qu'un patch doit être simulé, l'image d'entrainement est totalement ou partiellement scannée. Le résultat est obtenu d'autant plus rapidement que le sous-domaine de l'image d'entrainement parcouru est petit.

Table 4 Temps de calcul selon la fraction scannée de l'image d'entrainement. Le temps de référence T_0 correspond au temps de simulation lorsque toute l'image d'entrainement est scannée. Taille de template : 19×19. Taille du patch : 5×5. Chemin de visite régulier

Fraction	100%	80%	66%	50%
Temps de calcul	T_3	0.60 T_3	0.34 T_3	0.10 T_3

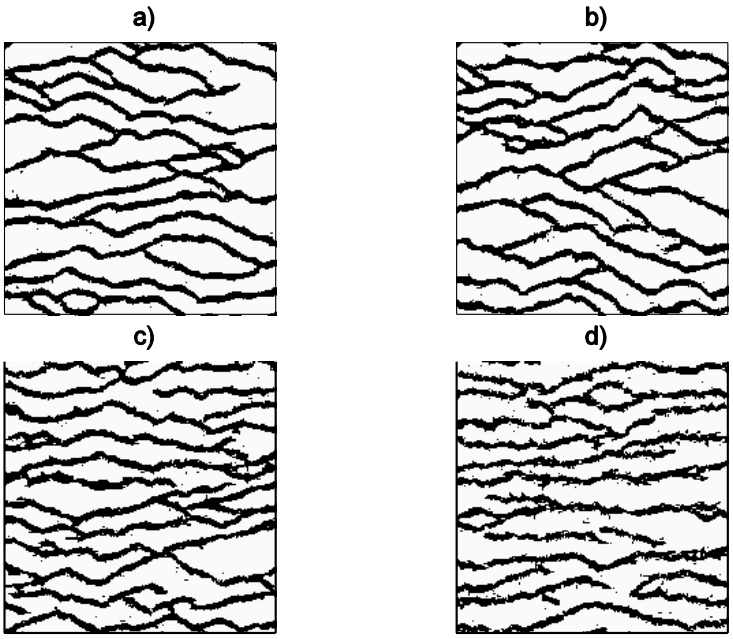


Figure 39 Réalisations simulées en ne scannant que : a) 100%, b) 80%, c) 66%, d) 50 %, de l'image d'entrainement. Taille du template : 19×19. Taille du patch : 5×5. Chemin de visite régulier.

La Figure 39 montre l'impact de la réduction de la zone balayée de l'image d'apprentissage. La continuité latérale et la connectivité des chenaux sont mieux restituées lorsque l'on parcourt une partie importante de l'image d'entrainement. Dans l'exemple ci-dessus, nous recommandons d'explorer au moins 66% de l'image d'entrainement. Les chenaux simulés sont alors relativement cohérents avec ceux figurant sur l'image d'entrainement. En outre, le temps de calcul est réduit à 34% de celui obtenu lors du balayage complet de l'image d'entrainement (Table 4).

En conclusion, pour l'exemple des chenaux étudié jusqu'à ce point, les deux stratégies décrites ci-dessus pour accélérer le processus de détermination du meilleur motif sont plus ou moins équivalentes en termes de temps de calcul et en termes de résultats. En effet, on obtient quasiment la même chose lorsque la simulation est réalisée avec 30 classes et prend en compte la cohérence, et lorsque 66% de l'image d'entrainement est parcourue. Dans ces deux cas, le temps de calcul est réduit à environ un tiers du temps de référence. Nous aurions pu essayer aussi de profiter des deux stratégies en les combinant.

4.5. Mesure de la distance entre le template et les motifs

Le temps de calcul nécessaire pour comparer le voisinage du pixel actuellement visité avec les motifs stockés dans la base de données est un point clé de l'algorithme. Il est répété autant de fois qu'il y a de motifs et chaque comparaison prend d'autant plus de temps que le template est grand. La différence est quantifiée à partir d'une distance telle que présentée dans la section 4.3.2.

Nous étudions maintenant le potentiel de différentes mesures. Premièrement, nous calculons les différences point-à-point en utilisant soit le carré de la norme L_2 soit la norme L_1. Deuxièmement, nous proposons l'application de la transformée en ondelettes dans une étape préliminaire avant d'estimer les différences point-à-point des coefficients de la transformée.

4.5.1. Mesure de distance définie par les normes L1 ou L2

Rappelons quelques notations. V est le voisinage du pixel actuellement visité et v_i la valeur du pixel i dans V. T est un motif extrait de l'image d'apprentissage et t_i est la valeur du pixel i dans T. Lorsqu'on se réfère à la norme L_2, la distance d entre V et T est donnée par l'EQUATION 20. Le carré de la différence pixel par pixel est pondéré par ω. Ce poids est plus grand lorsque le pixel d'intérêt est plus proche du pixel central du template (voir paragraphe 4.3.2). Il est égal à zéro lorsque le pixel ne contient pas de valeur (Figure 32 et Figure 33).

Pour calculer la distance d à partir de la norme L_2 :

$$d = \sum_{i \in V} \omega_i (v_i - t_i)^2 \qquad \textbf{Equation 20}$$

Pour calculer la distance d à partir de la norme L_1 :

$$d = \sum_{i \in V} \omega_i |v_i - t_i| \qquad \textbf{Equation 21}$$

Un exemple de réalisations simulées avec ces distances est montré sur la Figure 41. Les résultats sont très similaires quelle que soit la norme (L_1 ou L_2) utilisée. Les réalisations ont la même apparence et les temps de calcul sont équivalents.

4.5.2. Transformée en ondelettes

Pour réduire le temps nécessaire pour calculer une distance entre un template et un motif, nous avons testé la transformée en ondelettes. Une étape préliminaire consiste à faire une décomposition en ondelettes discrètes de premier niveau sur les motifs extraits de l'image d'apprentissage. Nous utilisons des ondelettes, car elles fournissent des approximations sur les dessins par sous-échantillonnage et ont la capacité de détecter les bords. Selon l'application, différentes ondelettes peuvent être choisies. Nous avons sélectionné les ondelettes de Haar parce qu'elles sont simples, efficaces en temps de calcul et localisées dans l'espace. Dans ce cas, le processus de décomposition est équivalent à la convolution des motifs avec des noyaux simples, comprenant les valeurs 1 et -1 (Figure 40).

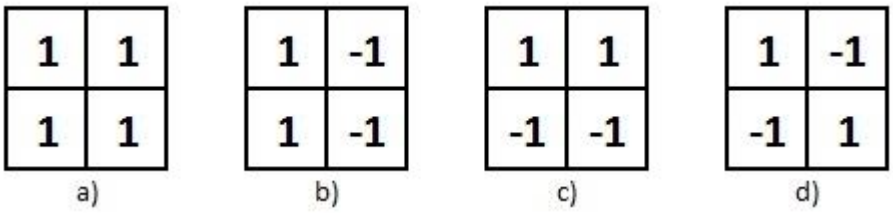


Figure 40 Noyaux de Haar. a) Basses fréquences. b) Hautes fréquences verticales. c) Hautes fréquences horizontales. d) Hautes fréquences horizontales et verticales.

La décomposition en ondelettes des motifs donne un ensemble de coefficients dont nous ne gardons que les principaux, qui sont $c = \{c_i\}_{i = 1 \dots n}$, n étant inférieur au nombre de pixels dans le template. La réduction du nombre de valeurs à comparer (juste les coefficients) contribue à réduire de manière significative la place mémoire.

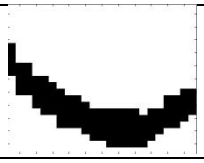
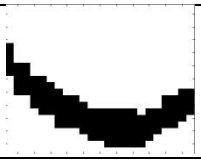
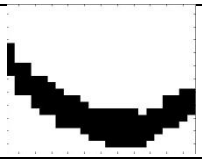
La première étape consiste à faire la décomposition en ondelettes de premier niveau de tous les motifs extraits de l'image d'entrainement. On a donc une bibliothèque de coefficients, chaque vecteur représentant un motif. Puis, de façon similaire à l'approche expliquée dans la section 4.4.1, ces vecteurs de coefficients sont classés en 20 catégories à l'aide de l'algorithme k-means.
Pendant la simulation, on visite les pixels un par un. Lors d'une itération donnée, nous déterminons le voisinage *V* du pixel courant et réalisons sa décomposition en ondelettes de premier niveau. Nous estimons alors la distance entre cette décomposition du voisinage (*i.e* les coefficients *c* dans *V*) et les 20 centres de gravité identifiés lors de la classification des décompositions des motifs extraits de l'image de la formation. Cela implique la définition de la distance suivante :

$$d = \sum_{i=1,n} \left(c_i^V - c_i^P \right)^2 \qquad \textbf{Equation 22}$$

Les coefficients *c* sont fournis grâce à une décomposition en ondelettes. Les exposants *V* et *P* indiquent si les coefficients correspondent au voisinage de *V* ou à la bibliothèque de coefficients de décomposition des motifs extraits *P*. À ce stade, une difficulté est de tenir compte du fait que *V* n'est pas entièrement défini : il peut inclure des pixels vides. Comme le montre la Figure 40, la décomposition en ondelettes de premier niveau implique que les coefficients c sont calculés à partir de 4 pixels. Par conséquent, nous ne tenons compte que des coefficients de décomposition calculés à partir d'au moins 3 pixels non vides. Si le calcul contient moins de 3 pixels alors le poids est mis à zéro. Il est à noter que si on ne tient compte que des coefficients calculés à partir de 4 pixels on enlève trop d'information et on dégrade alors l'image finale.

Afin d'illustrer le potentiel des ondelettes, on effectue un simple test. On sélectionne de façon aléatoire un motif dans l'image d'entrainement et on voit à quelle vitesse on peut retrouver ce motif en scannant toute l'image d'entrainement selon la mesure utilisée. Dans un premier test, on utilise la distance définie par le carré de la norme L_2 pondérée (Equation 20). Dans un deuxième test, on utilise la distance définie par la norme L_1 pondérée (Equation 21). Enfin dans un dernier test, on utilise la décomposition en ondelettes (Equation 22). Dans tous les tests on retrouve très bien le motif choisi aléatoirement. Les temps sont regroupés dans la Table 5. On voit qu'en utilisant la distance définie à partir de la transformée en ondelettes, le temps de calcul est 6 fois plus petit. La réduction de ce temps est un point clé, car l'étape de comparaison entre un voisinage et la bibliothèque de motifs est répétée des milliers de fois lors de la simulation.

Table 5 Temps de calcul pour trouver un motif dans l'image d'entrainement en fonction de la mesure de la distance. Taille de la fenêtre de voisinage: 19 x 19 pixels ; taille de l'image d'entrainement: 150 × 160.

Mesure	L2	L1	Ondelettes
Temps	0.046s	0.046s	0.007s
Motif sélectionné			

Sur la Figure 41, on compare 3 réalisations obtenues en utilisant les mesures décrites ci-dessus. La réalisation b) est simulée en utilisant le carré de la norme L_2 pondérée, la réalisation c) la norme L_1 pondérée et en enfin la réalisation d) en utilisant la transformée en ondelettes. Il n'y a pas beaucoup de différences entre les réalisations b et c, les chenaux sont bien reproduits ainsi que leur espacement. Sur la réalisation d) on voit qu'il y a plus de pixels noirs isolés et que les chenaux bien que correctement reproduits semblent plus rapprochés que dans l'image d'entrainement.

Concernant les temps de calcul pour la simulation avec transformée en ondelettes on est à 0.15 T_0 où T_0 est le temps de calcul de référence pour la simulation sans technique de clustering ou scanning partiel de l'image (Table 3, 1 classe). Un temps de calcul supplémentaire est nécessaire pour effectuer la transformée en ondelettes de la bibliothèque de motifs, qui est de 0.16 T_0.

En d'autres termes, si l'on utilise la norme L_2 sur les coefficients de la transformée en ondelettes comme mesure de distance et que l'on procède à une classification en 20 classes des motifs, on peut réduire le temps de calcul par deux par rapport au clustering 20 classes seul.

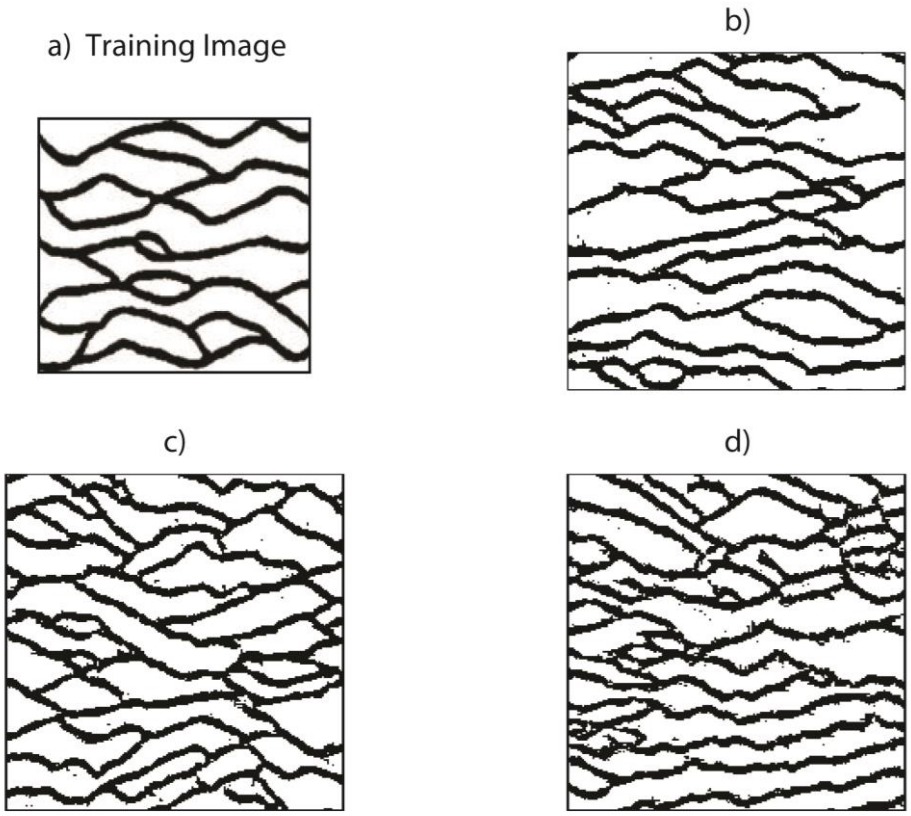


Figure 41 Réalisations simulées à l'aide des différentes définitions de distance. a): l'image d'entrainement ; b) la réalisation obtenue avec la norme L_2; c) la réalisation obtenue avec la norme L_1; d) La réalisation obtenue avec l'approche basée sur les ondelettes.

4.6. Quelques Résultats

Quelques essais sont présentés ci-après qui soulignent le potentiel et les limites de l'algorithme décrit dans le présent chapitre. Nous ne considérons que la version la plus simple de l'algorithme avec un scan complet de l'image d'entrainement et aucune classification préliminaire des motifs.

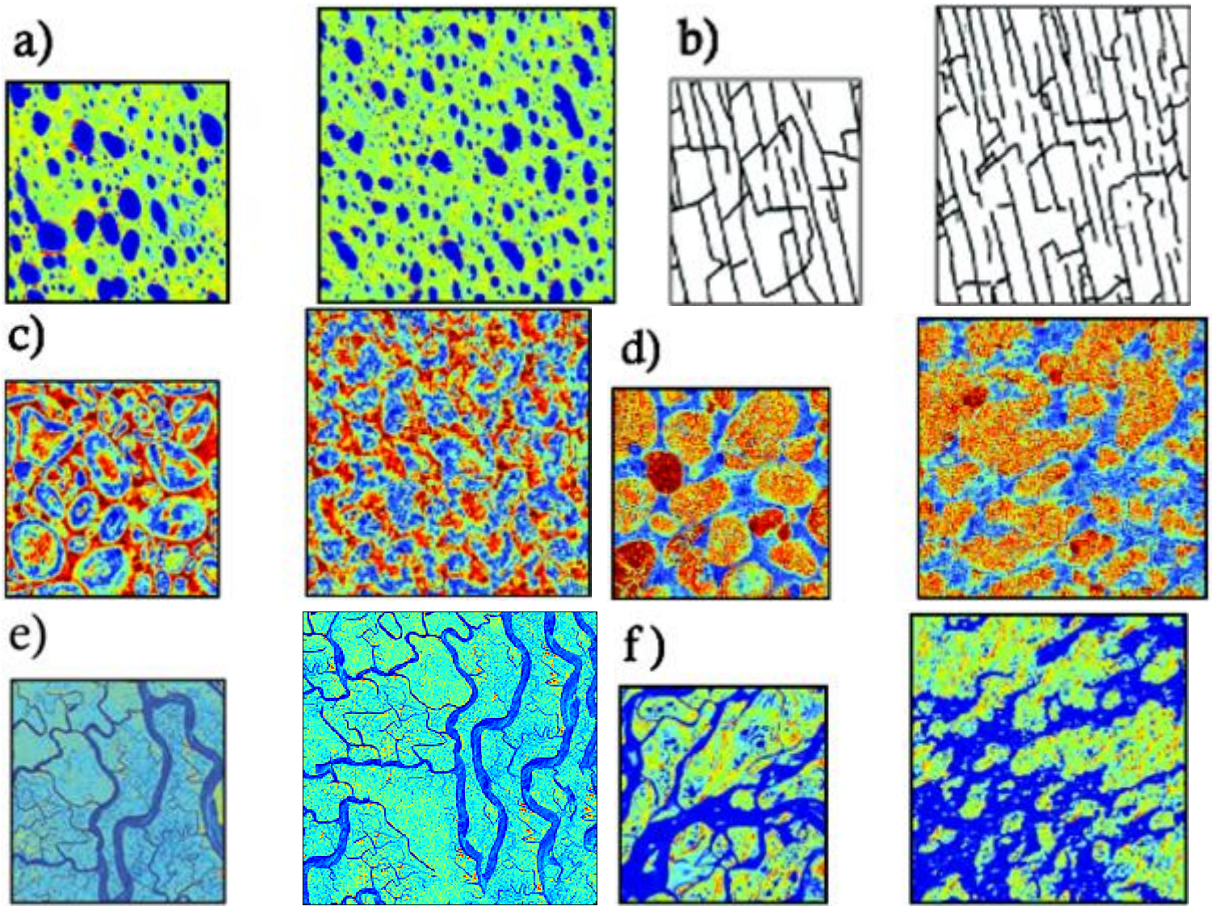


Figure 42 Les résultats obtenus par notre algorithme. Les petites images sont des images d'entraînement (150 × 160 pixels). Les plus grands sont les réalisations simulées (200 × 200 pixels).

Six exemples sont présentés dans la Figure 42. Certains d'entre eux représentent des objets à l'échelle microscopique comme les pores et grains (a, c et d) et des objets à l'échelles macroscopique comme des fractures ou des chenaux (b, e et f). Nous vérifions que le processus de simulation fonctionne bien pour les cas continus et discrets. Les réalisations générées sont semblables aux images d'apprentissage. Cependant, les résultats commencent à se dégrader lorsque les images d'entraînement comprennent de très grands objets tels que des gros grains ou des grandes chenaux. Une possibilité pour améliorer les résultats serait d'augmenter la taille du template. Cependant, il en résulterait une augmentation du temps de calcul et de la charge de mémoire.

4.7. Conclusion

Dans ce chapitre, nous avons développé un algorithme qui combine les deux concepts de simulation multipoints introduit en géosciences et de synthèse de texture utilisé dans l'infographie. En plus de sa simplicité et de son efficacité, l'algorithme proposé permet de simuler des objets géologiques avec des formes géométriques complexes. Le processus de simulation consiste à coller sur la grille de simulation un patch extrait d'une image d'apprentissage, selon un chemin en couronne jusqu'à ce que la grille soit entièrement simulée.

Des tests de sensibilité ont été effectués pour saisir les influences de 1) la taille de la fenêtre et du patch, de 2) la définition des poids, et de 3) le chemin de propagation lors de la visite des pixels. Nous avons montré que l'utilisation de grands templates permet de reproduire de grands objets. Cependant, cela induit une augmentation du temps de calcul et de la charge de mémoire. Nous avons également observé que l'application d'un noyau gaussien met l'accent sur l'importance des pixels centraux de la fenêtre de contexte et contribue à rendre les objets simulés plus semblables à ceux de l'image d'apprentissage. Enfin, nous avons montré que le respect d'un chemin de visite régulier au lieu d'un chemin aléatoire améliore la continuité et la connectivité de gros objets.

Nous avons montré que l'étape d'initialisation de l'algorithme pouvait être modifiée pour tenir compte des données dures tout en utilisant un chemin régulier de simulation par couronne. Le principe de base implique la construction d'une forme géométrique fermée reliant les données deux à deux. Par exemples, pour trois données on obtient un triangle, pour quatre données un carré, et pour cinq données et plus des polygones de forme un peu plus complexe. Les pixels traversés par les côtés de ces polygones sont simulés en premier, puis on crée des couronnes autour de ces formes en utilisant l'algorithme décrit ci-dessus.

Nous avons également étudié deux techniques de gestion de motifs extraits de l'image d'entrainement pour diminuer le temps de simulation. La première technique consiste à ne scanner qu'une partie de l'image d'entrainement lors de la simulation d'un patch. Plus on diminue la partie scannée, plus le temps de calcul diminue mais plus l'image finale est dégradée. La deuxième technique suit la même idée mais au lieu de ne scanner qu'une partie aléatoire de l'image d'entrainement on réduit intelligemment la partie scannée. On classe les motifs extraits par ressemblance et on leur attribue un représentant, *ie,* un motif proche représentatif de l'ensemble du groupe. On ne fait la recherche du meilleur motif que dans le groupe dont le représentant est le plus proche du voisinage du patch simulé. Plus le nombre de groupes est important plus le temps de calcul diminue, cependant comme pour la première technique on observe une dégradation des résultats de qualité de rendu. On introduit alors le concept de cohérence. Au lieu de ne scanner que le groupe du meilleur représentant, on scanne également les groupes à l'origine des pixels voisins. Cette approche permet d'augmenter la continuité et la connectivité des chenaux. L'image finale est alors bien plus semblable à l'image d'entrainement. En prenant les paramètres tels que l'image finale ne soit pas trop dégradée on arrive à diviser par 3 le temps de simulation.

De plus, nous avons testé différentes distances pour mesurer l'écart entre le voisinage du patch simulé et un motif extrait de l'image d'entrainement. Celle qui présente le plus d'intérêt pour la diminution du temps de calcul est la décomposition en ondelettes. La première étape consiste à appliquer une décomposition en ondelettes de premier niveau aux motifs de la base de données. Un motif est alors représenté par les coefficients de cette décomposition. Lors de la simulation, on procède à la même décomposition du voisinage, en ne gardant que les

coefficients de premier niveau et on les compare à ceux de la base de données. Comme il y a moins de coefficient à comparer, le temps de calcul est diminué. Considérant que les motifs ou les coefficients sont classés en 20 groupes, utiliser la décomposition en ondelettes permet de réduire encore le temps de calcul par 2.

Pour finir, six exemples sont testés pour montrer le potentiel et les limites de l'algorithme proposé. Nous avons vérifié que le processus de simulation fonctionne bien pour les deux cas continus et discrets, mais que la reproduction de très grands objets est toujours un défi.

Même si cette méthode montre des résultats encourageants, elle rencontre quelques difficultés à capturer de larges structures. De plus elle reste très coûteuse en temps de calcul. L'introduction d'une échelle intermédiaire devrait au moins en partie résoudre ces problèmes.

Chapitre 5

5. Algorithme Multipoint Multi-Echelles

Dans ce chapitre nous commençons par une bibliographie sur les méthodes multi-échelles de modélisation de réservoir. Puis nous détaillons l'adaptation de l'algorithme précédent pour introduire plusieurs échelles. Après une brève comparaison des résultats avec l'algorithme mono-échelle, nous nous focalisons sur le point clé de la méthode qui est l'obtention d'une image d'entrainement grossière. Enfin nous testons les techniques décrites précédemment pour accélérer les simulations.

5.1. Etat de l'art

Comme expliqué précédemment, l'idée derrière le développement d'un algorithme multi-échelles est de modéliser l'architecture du réservoir et de prendre en compte les différentes échelles de sa structure et des données qui sont collectées. Le réservoir peut contenir de grandes hétérogénéités, comme des méandres, et des petites hétérogénéités provenant de processus tels que le dépôt ou l'accrétion. En outre, les données sont disponibles à différentes échelles. Il existe quatre sources d'information : les mesures aux puits, l'étude sismique 3D, les données dynamiques de réservoir et l'interprétation géologique de l'architecture. Les mesures aux puits sont les plus précises, mais les moins nombreuses. Au contraire, les données sismiques sont disponibles sur tout le réservoir mais leur niveau de précision est plus faible. Lors de la construction du modèle de réservoir, il faut prendre en compte le fait que chaque élément d'information a sa propre caractéristique et échelle. L'intégration des données sismiques a été beaucoup étudiée dans le cadre des méthodes géostatistiques à deux points : Deutsch et al. (1996) utilisent un cokrigeage, Behrens et Tran (1998) utilisent la simulation gaussienne séquentielle avec un krigeage par bloc, Lee et al. (2000) utilisent une approche hiérarchique basée sur les champs de Markov aléatoires (Markov Random Fields). Toutes ces approches font l'hypothèse d'une relation linéaire entre les échelles, ce qui n'est valable que pour des variables additives.

Au cours de la dernière décennie, les algorithmes multipoints ont connus un enthousiasme croissant de par leur habilité à modéliser des architectures géologiques complexes telles que les méandres. Pour capturer de façon efficace les grandes hétérogénéités du réservoir, c'est l'approche multi-grilles qui a été généralement utilisée (Strebelle, 2002 ; Straubhaar, *et al.*, 2011 ; Kolbjørnsen, *et al.*, 2014 ; Straubhaar et Malinverni, 2014 ; Strebelle et Cavelius, 2014). Le concept multi-grilles est un sous-échantillonnage de grilles, de la plus fine vers la plus grossière. Les sous-grilles et les template ne sont que des versions espacés de la grille et du template initiaux. S'il y a n_g sous-grilles, à la grille k, les pixels pris en compte sont espacés de 2^{ng-k} pixels. Dans ce schéma, les pixels d'une sous-grille grossière appartiennent aux sous-grilles de plus haute résolution. Cette approche multi-grilles est illustrée Figure 43 dans le cas de trois grilles. Les templates associés aux grilles sont aussi montrés dans le cas où leur taille est de 3×3 pixels.

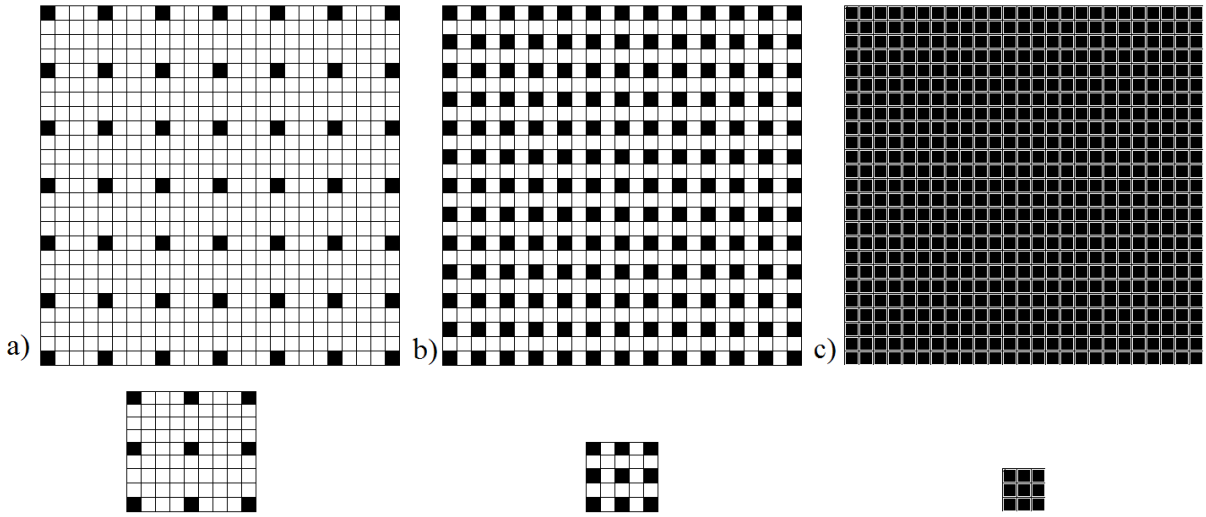


Figure 43 Illustration des sous-grilles et des sous-templates de l'approche multi-grilles. En noir ce sont les pixels qui appartiennent à la grille et qui sont utilisés. En blanc ce sont les pixels inutilisés.

Dans la même idée, certains auteurs se réfèrent à la simulation multi-échelles au lieu de simulation multi-grilles. Elle a été introduite en premier en infographie (De Bonet, 1997 ; Wei et Levoy, 2000 ; Ashikmin, 2001 ; Han, *et al.*, 2008). Au lieu de sous-échantillonner l'image d'entrainement, les valeurs de grille secondaire sont calculées à partir de celles de la grille de résolution plus fine. Ce calcul repose généralement sur l'interpolation bi-cubique en deux dimensions et sur l'interpolation tricubique en trois dimensions. En infographie, De Bonet (1997) a introduit l'idée de décomposer l'image d'entrainement en plusieurs sous-images de résolution inférieure, et d'utiliser les images de basse résolution pour contraindre la synthèse de l'image plus fine. Les techniques de Wei et Levoy (2000) pour l'infographie sont basées sur le même principe. L'image d'entrainement est décomposée en pyramide d'images de résolution décroissante, et l'image finale est simulée par résolution croissante. Le premier niveau de la réalisation finale est simulé avec le niveau le plus grossier de la pyramide des images d'entrainement. Lors de la simulation d'un niveau plus fin de la réalisation finale, on tient compte de la grille de simulation précédente. Un autre travail remarquable sur la synthèse de texture multi-échelles est la synthèse de l'image du continent, avec zoom de Han *et al.* (2008). La variété des zooms exigerait en entrée de simulation une quantité impossible d'images de différentes résolutions. Dans leurs travaux, l'image est construite à partir d'un graphe d'exemples de différentes images de résolution liées par des liens pondérés, réfléchis et orientés. En raison de l'approche multi-échelle, la simulation prend en compte les différentes échelles lors du choix des candidats. Il ne nécessite que quelques exemples en entrée.

Le principal avantage de ces approches est l'effet de dilatation du voisinage. Un petit template dans un schéma multi-échelles apporte plus ou moins autant d'information qu'un gros template dans un schéma mono-échelle. Cela permet d'améliorer la capture de grosses hétérogénéités avec de plus petits templates, ce qui rend la simulation plus efficace en termes de temps de calcul.

Le potentiel de l'approche multi-échelles, décrite en infographie, commence à être étudié en géosciences depuis peu.

Tahmasebi et Caers (2014) ont développé une extension de CCSIM dans une version multi-échelles appelée MS-CCSIM. L'image d'entrainement est décomposée en une pyramide d'images de résolution décroissante à chaque niveau. Cette pyramide d'images est utilisée pour réduire la taille de l'aire scannée dans l'image d'entrainement. Le domaine de la recherche à

l'échelle grossière est réutilisé à une échelle plus fine. Le domaine à l'échelle plus fine est centré sur le domaine à l'échelle plus grossière.

Un workflow intéressant a été proposé par Li et Caers (2011) pour un modèle hiérarchique de réservoir chenalisé. La première étape consiste à modéliser les grandes structures architecturales du réservoir à partir des données sismiques (emplacements des chenaux), puis à l'aide d'une technique multipoints, ils simulent les nappes de schiste le long des surfaces de ces chenaux. La deuxième étape de simulation consiste à ajouter des chenaux individuels dans les grands emplacements précédents, puis à re-simuler des nappes de schiste le long de ces plus petits chenaux. La dernière étape consiste à simuler les lithofaciès avec une simulation séquentielle gaussienne dans les petits chenaux.

L'approche multi-échelle est également bien connue dans la simulation basée sur les ondelettes. En infographie, Crouse et al. (1998) ont étudié les dépendances de coefficients d'ondelettes à différentes échelles pour la modélisation statistique des images. Des études montrent que la transformée en ondelettes discrètes (DWT) peut identifier les échelles dominantes et donner les relations entre les échelles (Aradhye et al, 2003 ; Droujinine, 2006), mais aussi gérer l'intégration multi-échelles de données (Gloaguen et Dimitrakopoulos, 2009). En 2012, Chaterjee *et al*. utilisent l'analyse par ondelettes dans la simulation par patch afin de réduire la dimension de la base de données des motifs.

Nous proposons ici, un algorithme multi-échelles et multi-résolutions qui est l'extension de l'algorithme précédemment étudié chapitre 4.

Dans la première partie, nous détaillons étape par étape l'algorithme multi-échelles. Dans la deuxième partie, une comparaison des algorithmes mono et multi-échelles montre l'amélioration apportée par l'introduction d'une seconde échelle. La troisième partie est consacrée à la représentation de l'image d'apprentissage à l'échelle grossière. Enfin, dans la quatrième partie, nous testons les deux techniques précédemment employées pour accélérer l'algorithme et comparer les résultats en temps CPU ainsi que la qualité de l'image finale synthétisée.

5.2. Algorithme

L'algorithme décrit dans le chapitre précédent atteint ses limites lorsque les hétérogénéités sont de grande taille. Il faut alors prendre de grands templates pour pouvoir capturer ce hétérogénéités ce qui implique un temps de calcul conséquent et des résultats assez décevants en terme de qualité de reproduction de l'image d'entrainement. Le but, ici, est de créer une échelle grossière, d'une résolution plus faible, donc rapide à synthétiser et qui rend compte des tendances des hétérogénéités. Dans un deuxième temps, on passe à l'échelle fine pour ajouter des détails à l'image simulée au préalable à l'échelle grossière. La synthèse à l'échelle fine ne nécessite alors pas de grands voisinages puisque que la forme générale des grandes hétérogénéités a été capturée au préalable dans l'image grossière. La synthèse est donc plus rapide.

L'algorithme de synthèse multi-échelles se déroule comme suit. Premièrement il faut définir les échelles selon les différents niveaux de résolution souhaités ou imposés, du plus bas au plus élevé. Il faut autant d'images d'entrainement que d'échelles. Deuxièmement on simule les réalisations de la plus grossière à la plus fine, en tenant compte à chaque niveau de tous les niveaux inférieurs. Ici, l'algorithme est expliqué sur deux échelles, pour simplifier. Il y a donc deux images d'entrainement, l'une pour l'échelle grossière, et l'autre pour l'échelle fine.

La première étape de l'algorithme consiste à synthétiser l'image grossière à partir de l'image d'entrainement grossière à partir de l'algorithme mono-échelle. Une fois la simulation faite, cette image ne change plus. La deuxième étape consiste à synthétiser l'image fine à partir de l'image d'entrainement fine en tenant compte de l'image grossière que l'on vient de synthétiser. Cette méthode est décrite ci-dessous pour la simulation sans contrainte.

Le lien entre les deux échelles est fait par la structure du template utilisée pour simuler la grille fine. Ce template contient les deux templates fins et grossiers, il contient donc dans une première partie les voisins de la grille fine et dans une deuxième partie les voisins de la grille grossière. Ils sont stockés comme une paire, comme le « dual-template » introduit par Arpat et Caers (2007) pour leur simulation multigrille. La recherche du motif le plus proche sur la grille fine est guidée par les deux échelles en comparant simultanément le template dans les deux images d'apprentissage.

Dans cette deuxième simulation, la première étape est d'initialiser la grille fine en sélectionnant un patch à coller au centre, mais pas au hasard: le choix du motif est contraint par le patch localisé au même endroit sur la grille grossière. Ensuite, nous remplissons la grille fine, couronne par couronne jusqu'à ce que la grille soit entièrement simulée. Une couronne est définie à un stade donné et comprend les mailles de grille qui n'ont pas déjà été simulées, et qui entourent immédiatement les mailles déjà simulées. La première couronne est celle autour du motif initial collé au centre. Une fois que cette première couronne est simulée, nous construisons la seconde autour de la première et ainsi de suite jusqu'à ce que la grille soit entièrement simulée. Pour toutes les mailles d'une couronne donnée, on compte le nombre de voisins contenant une valeur. Puis, on visite ces mailles en partant de celles qui a le plus de voisins simulés à celle qui en a le moins.

L'algorithme est assez proche de celui présenté plus haut pour l'approche mono-échelle. Les différences apparaissent dans la construction de la base de motifs et dans la comparaison des voisinages.

Seule la simulation à l'échelle fine est détaillée ici. On rappelle que I_c désigne l'image grossière synthétisée en utilisant l'algorithme mono-échelle à partir de l'image d'apprentissage Ti_c.
I_f est la réalisation fine à simuler à la fois à partir de l'image d'entrainement fine Ti_f, de l'image d'entrainement grossière Ti_c et de l'image déjà simulée I_c. p_f est le patch de pixels à simuler sur I_f à l'itération en cours, il est centré sur la maille fine i.
w est la fenêtre de contexte. $w(i)$ contient à la fois les voisins sur la grille fine $wf(i)$ et les voisins sur la grille grossière $wc(i)$. La fenêtre fine est centrée sur la maille i et la fenêtre grossière est centrée sur la maille grossière qui contient la maille i. Les étapes successives de la deuxième simulation de l'algorithme sont les suivantes.

Étapes de l'algorithme:

- Initialisation de I_f, guidée par l'image grossière déjà simulée I_c. La fenêtre de contexte ne contient que les voisins à l'échelle grossière. Ainsi, le motif choisi a sa partie de résolution grossière correspondant aux voisins sur la grille grossière au centre de I_c.
- Jusqu'à ce que toutes les mailles sur la grille fine soient visitées une fois:
    - Compte tenu des mailles déjà simulées sur la grille fine, identifier la couronne actuelle comme l'ensemble des mailles ayant déjà au moins un voisin simulé.
    - Trier les mailles de la couronne de celles qui a le plus de voisins à celles qui en a le moins. (Seuls les voisins sur la grille fine sont pris en compte, car sur la grille

grossière toutes les mailles sont simulées. Donc les mailles de la grille fine ont toutes le même nombre de voisins sur la grille grossière)

- On boucle sur les mailles selon le classement établi
    - Identifier le voisinage *w(i)* de la maille courante *i* selon le template. Il contient une partie fine avec les voisins sur la grille fine, et une partie grossière avec les mailles grossières correspondantes.
    - Comparer *w(i)* à la bibliothèque de motifs de référence recueillis dans les images d'entrainement et choisir celui qui est le plus proche de *w(i)*, la distance *d* entre les motifs et *w(i)* étant définie à partir d'une métrique. Notre choix par défaut est la norme L_2 pondérée par des poids en fonction de la position des mailles par rapport à la maille centrale (voir Figure 33).
    - Le patch central est alors extrait de la partie fine du motif sélectionné et collé sur *I* dans le groupe de pixels correspondant centré sur *i*.

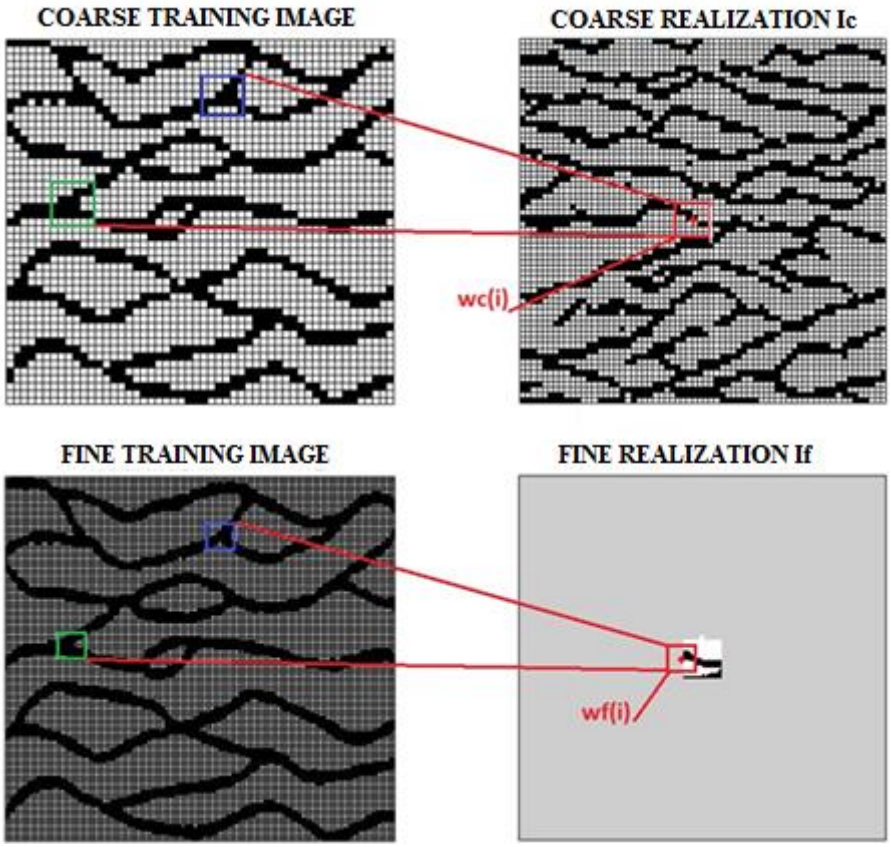


Figure 44 En haut à gauche, image d'entrainement grossière TI_c. En haut à droite, image synthétisée grossière I_c. En bas à gauche, image d'entrainement fine TI_f. En bas à droite, image synthétisée fine I_f. En bas à droite, pour le pixel *i* sur I_f, le pixel équivalent sur I_c est également marqué +, donc on compare le template en rouge sur la droite à la fois avec les motifs de la grille grossière et avec les motifs de la grille fine sur la gauche (vert et bleu).

Jusqu'ici on scanne toute l'image d'entrainement lorsque l'on simule un pixel. D'autres stratégies sont envisagées dans la section 5.5.

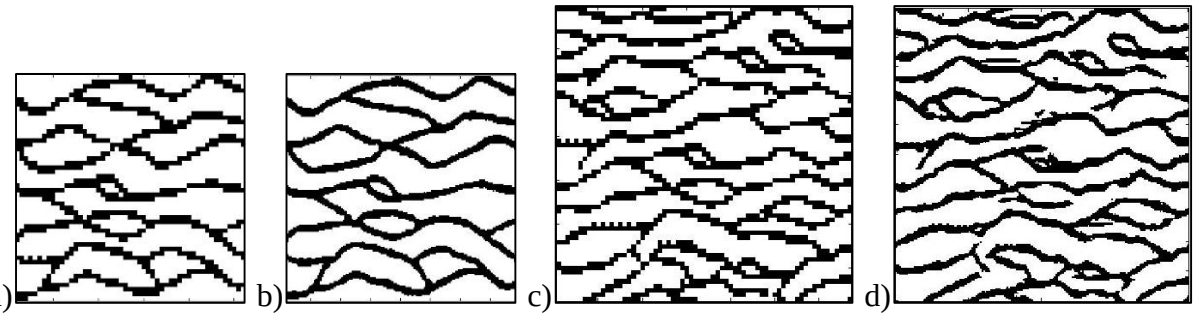


Figure 45 Simulation multi-échelles avec deux échelles. a) l'image d'entrainement grossière. b) l'image d'entrainement fine. c) Réalisation à l'échelle grossière. d) Réalisation à l'échelle fine. L'image d'entrainement grossière est obtenue à partir de l'image d'entrainement fine en attribuant à la maille grossière la valeur de la médiane du bloc de mailles fines correspondantes. Le template utilisé pour simuler la réalisation à l'échelle grossière comprend 9 × 9 mailles. Pour la simulation à l'échelle fine, la partie grossière du template comprend 3 × 3 mailles et la partie fine du template comprend 9 × 9 mailles.

Un exemple est illustré sur la Figure 45. Dans ce cas, l'image d'entrainement à l'échelle grossière est obtenue à partir de l'image d'entrainement à l'échelle fine. Pour chaque maille grossière, on calcule la médiane des mailles fine à l'intérieur. L'image d'entrainement grossière est constituée de 53 × 53 mailles. La réalisation à l'échelle grossière est obtenue en utilisant un template 9 × 9 et un patch 3 × 3. Il comprend 67 × 67 mailles. Dans ces conditions, le temps de simulation est inférieur à une seconde. La réalisation à l'échelle grossière semble plutôt bonne : les chenaux sont connectés et continus. Ensuite, on simule la réalisation fine (Figure 45d) conditionnellement à la réalisation à l'échelle grossière (Figure 45c). Le dual-template est construit à partir d'un template fin de 9 × 9 mailles, et d'un template grossier de 3 × 3 mailles. La réalisation fine respecte la tendance donnée par la réalisation grossière et reproduit les objets décrits par l'image d'entrainement fine. L'avantage de ce procédé multi-échelles est que l'on utilise un template de taille réduite pour reproduire l'image d'entrainement. Comme le template est petit, le temps de calcul diminue de manière significative par rapport à l'algorithme mono-échelle. Si l'on compare les réalisation représentées sur la Figure 29d et la Figure 45d, le temps de calcul est divisé par 7.

5.3. Comparaison des résultats à une et deux échelles

Tout d'abord, l'algorithme multi-échelles améliore la représentation des grandes structures. En effet, la synthèse multi-échelles permet de représenter de grandes hétérogénéités à l'échelle grossière avec un template plus petit (car la résolution est plus basse) et les détails sont ajoutés à l'échelle fine avec de même un petit template (car les grandes hétérogénéités sont déjà en place).

L'échelle grossière est une échelle de faible résolution, les images d'entrainement grossières sont (ici) 9 fois plus petites que l'image d'entrainement fine. Un pixel à l'échelle grossière représente les valeurs du bloc de 9 pixels adjacents à l'échelle fine. Les grandes hétérogénéités peuvent être capturées avec un template plus petit parce que l'image est plus petite. Les améliorations sont clairement visibles sur des structures telles que des rivières ou des matrices granulaires (voir Figure 46).

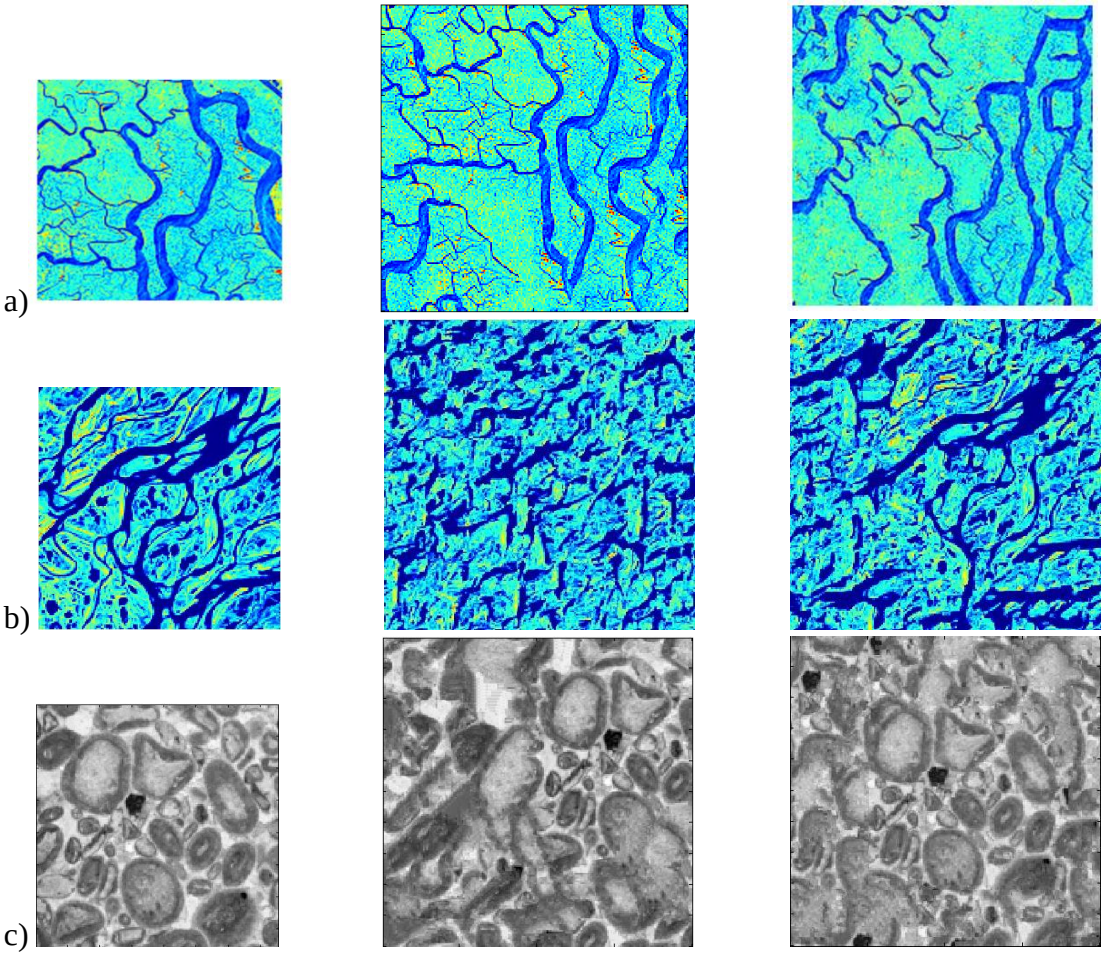


Figure 46 Comparaison de l'algorithme mono-échelle avec l'algorithme à deux échelles. Dans les colonnes de gauche à droite: Image d'entrainement ; Réalisation mono-échelle ; Réalisation multi-échelles. a & b) Simulation mono-échelle : Taille de template 21 × 21 et Taille de patch 3 × 3. Simulation deux échelles : Taille du template grossier 15 × 15 et Taille de patch 3 × 3, Taille du template fin 9 x 9, Taille du template grossier associé 3 x 3, et taille du patch 3 × 3. c) Simulation mono-échelle : Taille de template 21 × 21 et taille de patch 3 × 3. Simulation à deux échelles : taille du template grossier 15 × 15 et taille du patch 3 × 3, Taille du template fin 15 × 15, taille du template grossier associé 5 x 5, et taille du patch 3 × 3.

La Figure 46 montre les améliorations apportées par une approche multi-échelle. Dans la première colonne on trouve les images d'entrainement. Dans la deuxième colonne, on trouve les réalisations générées avec l'algorithme mono-échelle (chapitre 4). Dans la troisième colonne, on trouve les images construites par la méthode multi-échelles. Les deux premières images (lignes a et b) sont des photos satellites des rivières avec de grands méandres, la troisième image est une photo d'un morceau de quartz sous microscope.
On voit que dans la deuxième colonne, les réalisations de l'algorithme à une échelle ressemblent aux images d'entrainement mais présente des artefacts comme des lignes horizontales (a. 2ème colonne) ou diagonales (c. 2ème colonne). Pour diminuer le temps de calcul et améliorer la représentation des images on introduit une deuxième échelle de résolution plus basse. La troisième colonne montre les réalisations fines du processus multi-échelle. Les chenaux sont visibles et certains sont reliés. Clairement les réalisations finales semblent provenir d'un même processus de formation que les images d'apprentissage.

D'autre part, cet algorithme améliore également le temps de calcul. En comparaison avec la simulation à une échelle, dans laquelle la taille du template est de 21 × 21, la simulation à deux échelles utilise pour la simulation grossière un template de taille de 9 × 9 pixels, et pour la simulation fine un template fin de 9 × 9 et un template grossier associé de 3 × 3. Le temps de

calcul est divisé par 7. La simulation à l'échelle grossière tourne en une ou deux secondes, et la simulation fine échelle en 68 secondes.

Table 6 Comparaison des temps de simulation entre l'algorithme mono-échelle et l'algorithme multi-échelles

	Figure 46 a)	Figure 46 b)	Figure 46 c)
Simulation mono-échelle	T_0 = 420 sec	T_0	T_1
Simulation à 2 échelles	T_0 /6	T_0 /6	T_1/3.6

Le point intéressant de cet algorithme est que la réalisation fine est conditionnée par la réalisation à l'échelle grossière. Comme on peut le voir sur la Figure 47, l'image fine suit les tendances données par l'image grossière. Les images grossières donnent l'emplacement des méandres, ainsi que leur forme. La simulation fine vient ajouter des détails, sans modifier les grandes structures de l'image.

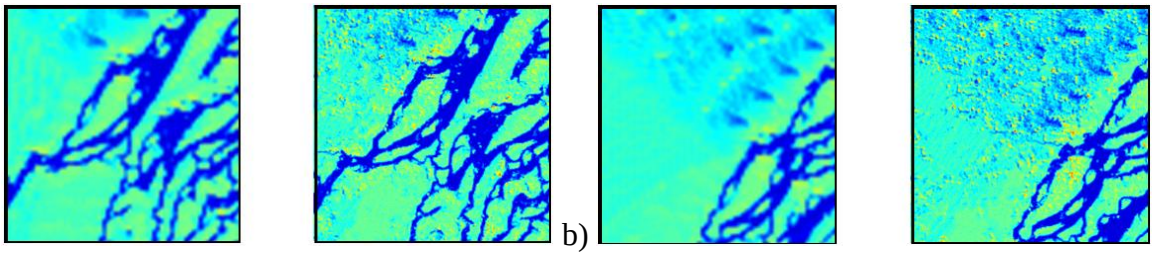


Figure 47 Impact de la réalisation à l'échelle grossière sur la réalisation à l'échelle fine. Sur les figures a) et b) à gauche se trouvent les réalisations à l'échelle grossière et à droite les réalisations à l'échelle fine. Les réalisations à l'échelle grossière sont simulées avec un template de taille 13×13 et un patch de 3×3 pixels. Les réalisations à l'échelle fine sont simulées avec l'algorithme multi-échelle, un template de taille 15×15 et un patch de taille 5×5.

5.4. Origine de l'image grossière

Nous nous concentrons ici sur deux images décrivant la même structure, mais à des échelles différentes. Par exemple, à l'échelle grossière les chenaux apparaissent mais pas les différentes couches de dépôts de sédiments et à l'échelle fine tout est visible.

5.4.1. Cas d'une propriété continue

Cette section est consacrée au calcul de l'image à l'échelle grossière à partir de l'image à l'échelle fine. On considère une propriété continue telle que la perméabilité. Les images grossières viennent des images de résolution fine filtrées. Nous avons testé différents types de filtrage: moyenne, médiane, et NL-means. Nous montrons leur impact sur l'image de synthèse.

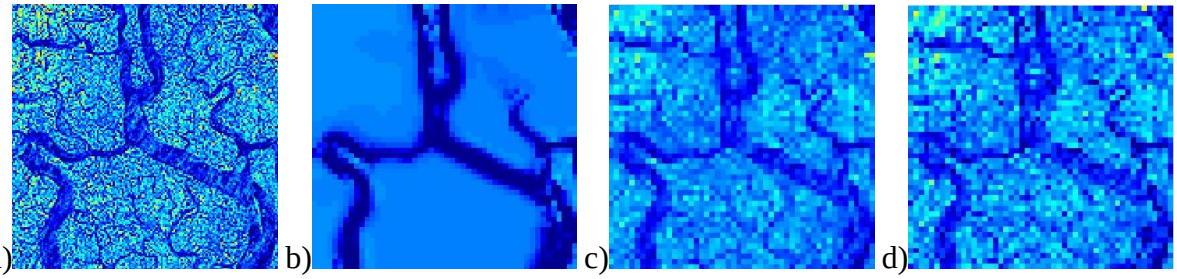


Figure 48 Comparaison des différentes images secondaires en fonction de leur voie d'acquisition. a) l'image d'origine. b) l'image filtrée avec NL-means, puis la moyenne. c) l'image moyennée. d) l'image après la prise de la médiane.

5.4.1.1 Moyenne et Médiane

L'idée est que l'image à l'échelle grossière est la même que l'image à l'échelle fine, mais à une résolution inférieure. La méthode de la moyenne (médiane) consiste simplement à prendre la moyenne (médiane) d'un bloc de pixels de la grille fine et à l'affecter au pixel central de la grille et à éliminer les autres pixels du bloc.
L'équation ci-dessous donne la valeur du pixel $TI_c(i, j)$, appartenant à l'image grossière, comme étant la moyenne des pixels de $TI_f(p, q)$, appartenant à l'image fine.
Soient NX_f et NY_f, les nombres de pixels suivant les axes x et y pour l'image fine, et NX_c et NY_c, les nombres de pixels suivant les axes x et y pour l'image grossière. a est la différence de résolution entre l'image fine et l'image grossière. Alors, on a :

$$NX_c = {NX_f}/{a} \qquad et \qquad NY_c = {NY_f}/{a}$$
$$\forall i \in [1, NY_c], \forall j \in [1, NX_c], \qquad TI_c(i,j) = \frac{1}{a^2} \sum_{p=a\times(i-1)}^{a\times i} \sum_{q=a\times(j-1)}^{a\times j} TI_f(p,q) \qquad \textbf{Equation 23}$$

Par exemple ici dans la Figure 48, a = 3, un seul bloc de grille sur neuf est maintenu et sa valeur est la moyenne des neuf pixels (Figure 48 c) ou la médiane des neuf pixels (Figure 48 d).

5.4.1.2 NL-means

L'algorithme Non Local means est un algorithme de traitement d'image pour débruiter des images. Contrairement aux filtres de lissage, cet algorithme ne met pas à jour la valeur d'un pixel selon une moyenne du voisinage mais plutôt en le moyennant sur l'ensemble des pixels de l'image qui ont le même voisinage. Chaque pixel est pondéré selon la distance entre son voisinage et celui du pixel mis à jour. L'algorithme NL-means a été publié par Antoni Buades *et al*. en 2005.

Considérant l'image *I*, le pixel *i* à simuler, et *v(i)* sa valeur, *NL[v](i)* est la valeur estimée de *v(i)*, donnée par :

$$NL[v](i) = \sum_{j \in I} w(i,j)v(j)$$
$$0 \le w(i,j) \le 1 \quad et \quad \sum_j w(i,j) = 1 \qquad \textbf{Equation 24}$$

La famille de poids $\{w(i,j)\}_j$ dépend du degré de similitude entre les pixels *i* et *j*. La similarité entre deux pixels *i* et *j* dépend de l'intensité des vecteurs de niveau de gris *v(Ni)* et *v(Nj)* où *Nk* est un voisinage carré de taille fixe et centré sur le pixel *k*. La similitude est mesurée avec une norme L_2 et les poids sont donnés par :

$$w(i,j) = \frac{1}{Z(i)} e^{\left[-\frac{\|v(Ni)-v(Nj)\|^2_{2,a}}{h^2}\right]} \qquad with \;\; Z(i) = \sum_j e^{\left[-\frac{\|v(Ni)-v(Nj)\|^2_{2,a}}{h^2}\right]}$$

h est le degré de filtrage. Il contrôle la décroissance de la fonction exponentielle, et donc la décroissance des poids.
L'algorithme a deux autres paramètres tout aussi importants qui pilotent le débruitage : *t* et *f*.

t est la zone scannée pour la recherche des voisinages similaires. *F* est la taille du voisinage *Nk* centrée sur *k*. Ils sont représentés sur la Figure 49 :

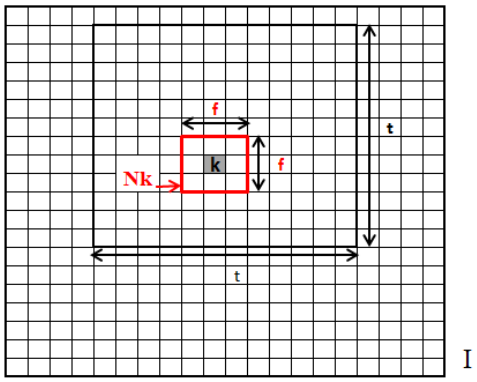


Figure 49 Illustration des paramètres de l'algorithme NL-means

La Figure 50 illustre la sensibilité au paramètre *h*, de l'image filtrée. Plus *h* augmente plus le filtrage est efficace et les plus petits détails sont effacés. Toutefois, avec un *h* supérieur à 100, l'image est trop floue, même la structure principale commence à être effacée.

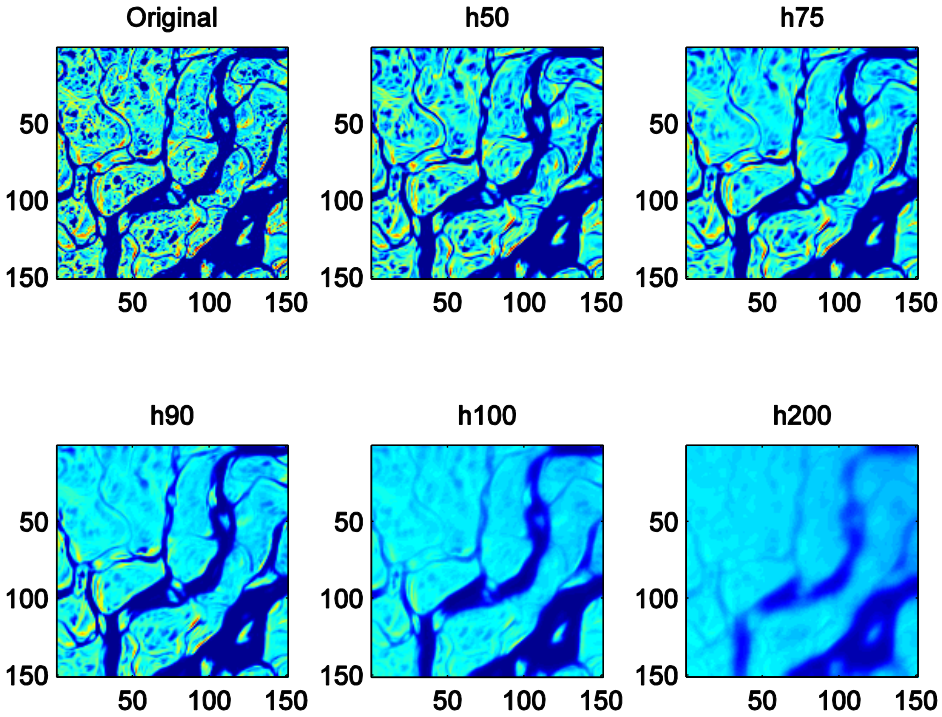


Figure 50 Photo satellite du delta du fleuve Lena. Comparaison de l'impact du degré de filtrage *h* sur le processus de débruitage. Les autres paramètres sont fixés : *t* est égal à 50 et *f* est égal à 7. En haut, de gauche à droite : La photo originale, la photo débruitée avec *h* égal à 50 et avec *h* égal à 75. En bas, de gauche à droite : images débruitées avec *h* égal respectivement à 90, 100 et 200.

5.4.1.3 Impact du filtrage sur l'image finale

Intuitivement, il semble normal qu'une image avec un contraste plus élevé donne de meilleurs résultats, quelle que soit la manière de calculer la distance entre deux motifs. Pour tester et vérifier cette idée, nous avons filtré une photo satellite bruitée montrant le delta du Gange (Figure 51 a) pour obtenir une image de contraste élevé (Figure 51 c). Ensuite, nous

procédons à la simulation mono-échelle avec les deux images d'entrainement différentes (filtrée et non filtrée). Les résultats (Figure 51 b et d) montrent que l'image synthétisée à partir de la photo filtrée donne une meilleure qualité dans le rendu final.

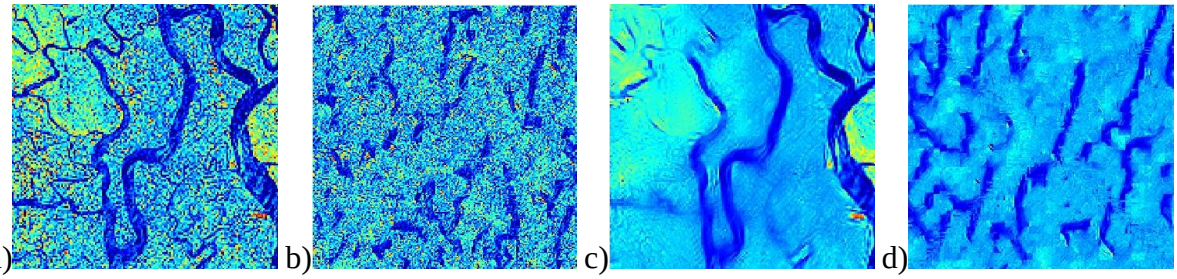


Figure 51 Influence du filtrage sur la simulation à une échelle. a) l'image d'entrainement originale b) l'image simulée à partir de l'image non-filtrée c) l'image d'entrainement filtrée d) l'image synthétisée à partir d'une image d'entrainement filtrée. Les deux réalisations sont simulées avec un voisinage de 21 x 21 et un patch de 3 x 3.

Nous avons montré précédemment que la réalisation à l'échelle grossière guide la réalisation à l'échelle fine. Donc une réalisation à l'échelle grossière qui représente bien les chenaux (leur structure, leur espacement, *etc.*), permet d'obtenir une bonne réalisation à l'échelle fine. Comme on vient de le montrer sur la **Figure 51**, à partir d'une image de faible contraste on ne parvient pas à recréer de chenaux, donc introduire une phase de filtrage à l'échelle grossière permet d'améliorer la qualité de l'image d'entrainement grossière et par conséquent d'améliorer le rendu de la simulation à l'échelle grossière.

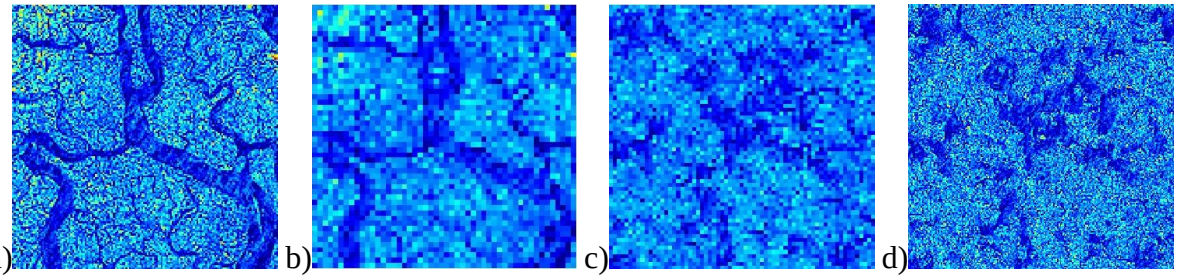


Figure 52 Impact du non-filtrage sur la simulation à deux échelles. a) l'image d'entrainement fine. b) l'image d'entrainement grossière, obtenue par moyenne de l'image fine c) la réalisation à l'échelle grossière avec un template de 9 x 9 et un patch de 3 × 3. d) la réalisation fine avec un template fin de 9 × 9 et un template grossier associé de 3 × 3 et un patch de 3 × 3

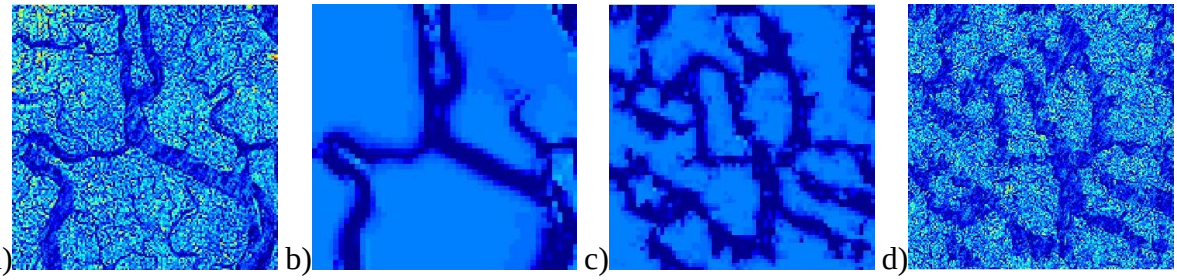


Figure 53 Impact du filtrage sur la simulation à deux échelles a) l'image d'entrainement fine. b) l'image d'entrainement grossière, obtenue par NL-means de l'image fine, $h = 100$, $f = 11$, $t = 50$. c) la réalisation à l'échelle grossière avec un template de 9 x 9 et un patch de 3 × 3. d) la réalisation fine avec un template de 9 × 9 et un template grossier associé de 3 × 3 et un patch de 3 × 3

Au cours du processus de simulation multi-échelles, la simulation mono-échelle à l'échelle grossière est essentielle, en ce qu'elle donne la tendance, la connectivité et la continuité

de grandes structures. Si la réalisation grossière n'est pas de bonne qualité, il en va de même pour la réalisation fine. L'algorithme mono-échelle est plus performant avec l'image d’entrainement de haut contraste.
Sur la Figure 52, l’image d’entrainement (Figure 52 a) a un faible contraste entre les chenaux de la rivière et la terre. L'image d’entrainement grossière (Figure 52 b) est obtenue simplement par moyenne de l’image fine. Donc elle a encore un faible contraste. La réalisation à l'échelle grossière (Figure 52 c) ne représente pas correctement la continuité des chenaux, ce qui conduit à la mauvaise qualité de la réalisation fine (Figure 52d).
Dans la Figure 53 a) nous gardons la même image fine d’entrainement, mais nous l’avons filtrée à l’aide de NL-means pour obtenir l'image d’entrainement grossière (Figure 53 b) plus contrastée. La réalisation grossière Figure 53c) est de meilleure qualité, les chenaux sont continus et connectés. La réalisation fine (Figure 53 d) suit les tendances données par la réalisation grossière, et représente mieux les caractéristiques de l’image d’entrainement fine.

5.4.2. Cas d’une propriété discrète

Dans cette section on se concentre sur les images représentant des propriétés discrètes, telles que des fractures. Ces images peuvent être très complexes, contenant plusieurs niveaux de fracturations et donc de fractures de tailles différentes. Comme les fractures impactent très fortement les écoulements à l’intérieur du réservoir, il est nécessaire de reproduire au mieux le réseau naturel.
L’image d’entrainement utilisée Figure 54 vient d’une carte de fractures dans une formation calcaire du champ de Karahayit dans l’ouest de la Turquie (Jafari et Babadagli, 2010). Sur l’image les fractures sont en noir (ce sont des zéros dans l’image), et la matrice encaissante en blanc (représentée par des 1).

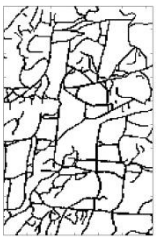


Figure 54 Image d’entrainement fine (125×192 pixels).

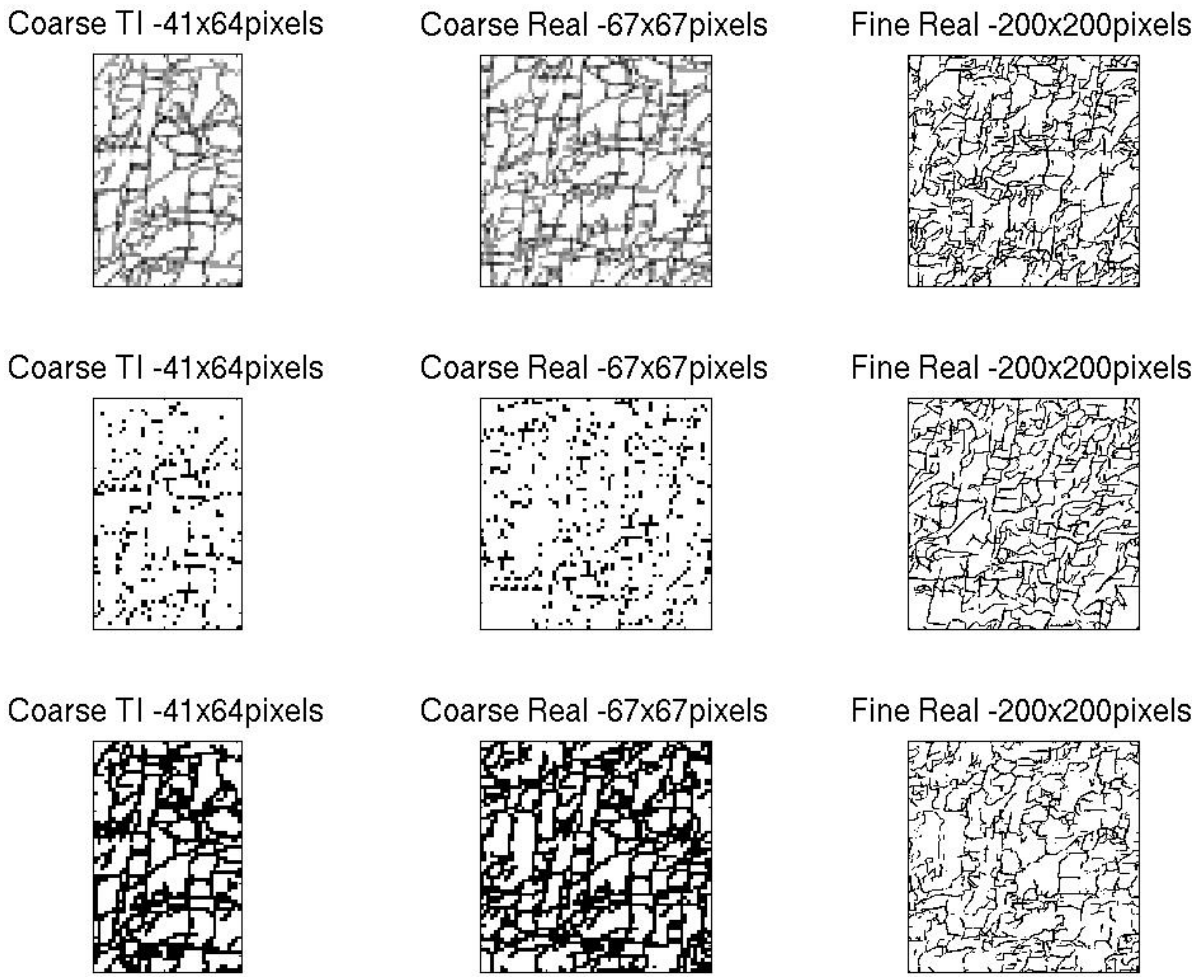


Figure 55 Impact de la technique utilisée pour obtenir l'image d'entrainement grossière. 1ère ligne : moyenne arithmétique. 2ème ligne : médiane. 3ème ligne : minimum. 1ère colonne : Les images d'entrainement grossières. 2ème colonne : Les réalisations grossières simulées à partir des images grossières de la 1ère colonne. 3ème colonne : Réalisations fines conditionnées par les réalisations grossières de la 2ème colonne. (Paramètres utilisés : différence de résolution entre les échelles 3×2 ; taille de template grossier 9×9 ; taille du patch 3×3 ; taille du dual template 13×13 fin, 3×3 grossier ; taille du patch 3×3 ; 3 classes)

Trois techniques ont été testées ici : la moyenne et la médiane, qui sont identiques à celles expliquées section 5.4.1 et la prise du minimum, qui est simplement le fait de prendre une occurrence de fracture sur le pixel grossier si le bloc de pixels fins est traversé par une fracture. La moyenne donne une image grossière avec des variations continues le long des fractures. La médiane donne plutôt un ensemble de points. Et la prise du minimum donne une version discrète de l'image obtenue par la moyenne (Figure 55, 1ère colonne).
Intuitivement on s'attend à ce que les prises de moyenne et du minimum donnent de meilleurs résultats car les images d'entrainement grossières obtenues ressemblent plus à celle d'origine et décrivent mieux le réseau de fractures. Au contraire, l'image d'entrainement grossière obtenue par la prise de médiane, ne ressemble pas du tout à l'image d'origine mais plutôt à un ensemble presque aléatoire de points. La 2ème colonne (Figure 55) semble confirmer cette intuition. Les réalisations grossières sont semblables aux images d'entrainement grossières à partir desquelles elles ont été simulées. Cependant les résultats de la 3ème colonne (Figure 55) inversent ces observations. La réalisation fine conditionnée par la réalisation grossière obtenue par moyenne, présente une densité de petites fractures trop élevée. Celle conditionnée par l'image grossière obtenue par prise de minimum, ne présente pas assez de connectivité entre

les fractures. Finalement c'est la réalisation fine conditionnée par la réalisation grossière obtenue par prise de médiane qui donne les meilleurs résultats.

5.5. Techniques de diminution du temps de calcul

On teste ici les méthodes testées en section 4.4 pour diminuer le temps de calcul.

5.5.1. Scan partiel de l'image d'entrainement fine

La première simulation mono-échelle à l'échelle grossière reste la même. Le temps de la simulation grossière mono-échelle est très court, et l'image d'entrainement grossière est petite. Nous n'avons donc pas besoin de réduire la partie scannée. On ne joue que sur la deuxième simulation, celle de la réalisation fine à partir de la réalisation grossière et de l'image d'entrainement fine.
On se concentre ici sur la simulation de la réalisation à l'échelle fine. Pour attribuer une valeur à un pixel de la grille fine, on ne scanne qu'une partie de la base de données des doubles motifs (les motifs qui comprennent une partie fine et une partie grossière).
Contrairement à l'algorithme mono-échelle, ici la partie scannée peut être réduite à 50% sans trop dégrader l'image finale. Une des raisons principales venant du fait que la simulation est contrainte par la réalisation à l'échelle grossière qui donne la continuité et la connectivité des chenaux.

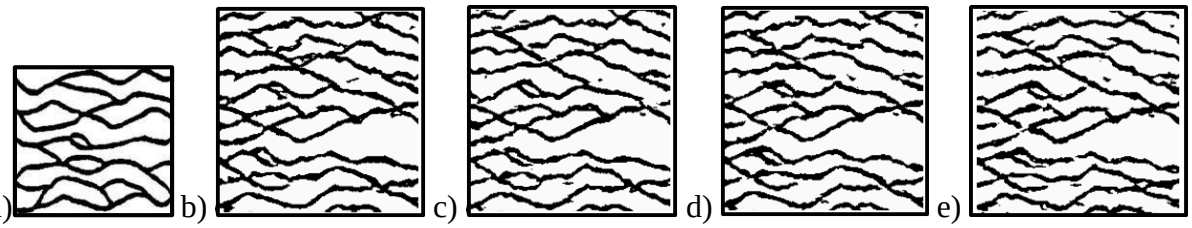


Figure 56 Impact de la proportion de l'image d'entrainement scannée au cours de la simulation a) l'image d'entrainement b) 100% c) 75% d) 50% e) 25%. Réalisation grossière simulée avec un template 9×9 et un patch 3×3. Réalisations fine simulées avec un dual-template 9×9, 3×3 et un patch 3×3.

Table 7 Comparaison du temps CPU de simulation avec des proportions différentes image numérisée de la formation

Proportion scannée	100%	75%	50%	25%
Temps de simulation	T_2=68sec	0.76 T_2	0.48 T_2	0.23 T_2

Sur la Figure 56, on peut voir l'effet de la proportion de l'image scannée au cours du processus de simulation. Ces images sont construites à partir de la même réalisation grossière. Elles sont simulées avec un template de 9 × 9 et un patch 3 × 3. Ensuite, lors de la simulation de la réalisation fine, nous scannons seulement une certaine proportion de la base de données de dual-templates pour trouver le meilleur motif. Cette proportion passe de 100% à 25% (Figure 56 b à e). Dans la Table 7, les temps de simulation relatifs sont donnés pour chaque simulation. Le temps de référence T_2 est celui de la simulation avec le scan total de l'image d'entrainement. Les temps sont proportionnels à la partie de l'image scannée. Le temps est divisé par deux lorsque la moitié de l'image d'apprentissage est balayée.

5.5.2. Kmeans clustering

La base de données est ici construite avec les motifs extraits des images d'entrainement à la fois grossière et fine. On rappelle qu'un template multi-échelles contient d'abord le template extrait de l'image d'entrainement fine et un deuxième template extrait de l'image d'entrainement grossière correspondant au template fin.
Nous regroupons ces templates multi-échelles en *k* classes et chaque classe est représentée par le template multi-échelle le plus proche de la moyenne des templates de la classe.
Pendant le processus de simulation, nous comparons le voisinage du pixel simulé avec les représentants des *k* classes et trouvons le plus proche. Finalement on ne compare voisinage du pixel visité qu'avec le groupe du représentant le plus proche.
La recherche du meilleur motif est limitée à ce groupe, conduisant ainsi à une accélération significative de la simulation.

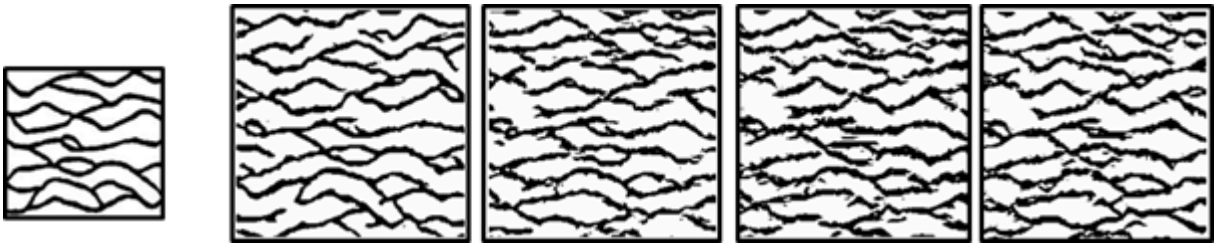


Figure 57 Impact du nombre de classes sur l'image synthétisée. a) l'image d'entrainement b) une classe c) 10 classes d) 20 classes e) 30 classes. La simulation à l'échelle grossière utilise un template 9 × 9 et un patch 3 × 3. La réalisation fine utilise un double template de 9 × 9 et 3 × 3, avec un patch 3 × 3.

Table 8 Comparaison du temps de calcul suivant le nombre de classes utilisées pour trier les dual-templates

Nombre de classes	1 classe	10 classes	20 classes	30 classes
Temps	T_2	0.43 T_2	0.32 T_2	0.19 T_2

Dans la Figure 57, nous montrons l'impact de la classification sur la qualité de la réalisation fine finale. Toutes les réalisations sont simulées avec le même ensemble de paramètres. Le seul changement est le nombre de classes dans le k-means. De toute évidence la réduction du nombre de motifs lors de la comparaison réduit la chance d'avoir le meilleur motif possible et devrait comme dans la section 4.4.1 dégrader la qualité de l'image fine finale. Toutefois nous pouvons voir que même avec 30 classes les chenaux sont encore continus et toujours connectés. Il y a plus d'interruptions brusques de chenaux, et plus de pixels isolés (bruits), et la réalisation finale est encore très similaire à l'image d'entrainement. Dans la Table 8, on rapporte les temps relatifs. La simulation avec 30 classes permet de diviser le temps CPU par 5 en comparaison avec la simulation sans classification.

5.6. Conclusion

Dans ce chapitre on a présenté un algorithme multipoint multi-échelles. Pour simplifier nous ne l'avons illustré que sur deux échelles, une fine correspondant à la résolution de l'image d'entrainement originale, et une grossière de résolution inférieure calculée à partir de l'image d'entrainement fine. L'algorithme se déroule alors en quatre étapes comme suit. Premièrement, on calcule une image d'entrainement grossière à partir de l'image d'entrainement fine. Deuxièmement, on simule la réalisation à l'échelle grossière en appliquant l'algorithme mono-

échelle. On utilise alors l'image d'apprentissage grossière. A partir de l'étape trois, on rentre dans le processus multi-échelles. Troisièmement, on construit la base de données multi-échelles à partir des motifs extraits des deux images d'entrainement. Et quatrièmement on simule la réalisation fine, à partir des images d'entrainement fine et grossière, conditionnée par la réalisation grossière que l'on vient de simuler. Des tests ont été réalisés pour mettre en évidence les avantages d'une approche multi-échelles en comparaison d'une approche mono-échelle. L'avantage principal indéniable est que l'algorithme multi-échelles capture les grandes hétérogénéités avec de plus petits templates, ce qui diminue considérablement les temps de calcul.

Un des points clé de la méthode multi-échelles, est l'obtention de l'image d'entrainement à l'échelle grossière, que ce soit pour des propriétés continues ou discrètes. Premièrement, on a étudié une image satellite d'une partie du delta du Gange particulièrement mal contrastée, pour mettre en évidence un problème qui peut arriver avec les variables continues. L'application d'un filtre avant la prise de moyenne, pour augmenter les contrastes et faire ressortir les structures principales de l'image, améliore la reproduction de ces grandes structures à l'échelle grossière et donc à l'échelle fine. Deuxièmement, on a étudié une image d'un réseau de fractures. On a vu que c'est la prise de médiane à l'échelle grossière qui permet d'obtenir une réalisation fine la plus semblable au réseau de l'image d'entrainement fine.

Pour terminer nous avons testé les techniques d'accélération décrites dans le chapitre précédent (Chapitre 4). Elles sont toutes deux basées sur la même idée de réduction du nombre de motifs à comparer lors de la simulation d'un patch. Dans la première approche, on ne scanne qu'une partie de la base de données, sélectionnée aléatoirement. Dans la seconde approche, on réalise d'abord une classification par classes de la base de données, et on ne scanne qu'une classe lors de la simulation du patch. Ces deux méthodes permettent de réduire significativement les temps de calcul. Cependant contrairement à la méthode mono-échelle, les réalisations finales sont beaucoup moins dégradées. En effet la réalisation à l'échelle grossière étant simulée très rapidement, il n'y a pas besoin de la modifier, elle reste donc de bonne qualité. Le point clé de l'approche multi-échelle est de simuler la réalisation fine avec comme contrainte la réalisation grossière. Si la réalisation à l'échelle grossière est de bonne qualité, elle conditionne très fortement la réalisation à l'échelle fine qui est alors de bonne qualité.

Conclusion Générale - Perspectives

La modélisation de réservoir est une étape importante dans l'exploration et l'exploitation d'un gisement d'hydrocarbures. C'est l'image du réservoir que l'on va construire à partir de toutes les données disponibles, et qui va servir à guider la gestion du champ. Vouloir obtenir l'image la plus réaliste et la plus exacte possible, le plus rapidement possible est alors compréhensible.

Nous avons proposé deux méthodes de simulation multi-échelles de réservoir. L'idée basée sur l'observation de l'architecture complexe des réservoirs et des différentes résolutions des données disponibles, était de copier dans un modèle cette structure à plusieurs échelles dépendantes les unes des autres, et bien évidemment d'en tirer partie en prenant en compte les grandes hétérogénéités aux échelles de plus basses résolutions pour diminuer les nombre de paramètres à caler.

Tout d'abord nous avons décrit la méthode de simulation séquentielle Gaussienne. C'est une technique qui s'inscrit dans la famille des méthodes géostatistiques à deux points. L'algorithme multi-échelles proposé permet de simuler une variable à l'échelle fine à partir d'une variable moyenne simulée au préalable à l'échelle grossière. Le point clé de ce passage d'une échelle à l'autre, est le calcul de la covariance croisée et de la covariance à l'échelle grossière, comme nous l'avons montré. L'un des avantages de cet algorithme est qu'il permet de modifier les réalisations au niveau des deux échelles simultanément ou séparément. La mise en place d'un cas synthétique et d'une boucle de calage d'historique a montré que la possibilité d'apporter des modifications sur les réalisations à l'échelle grossière s'avère particulièrement avantageuse. Cependant sur ce cas synthétique la comparaison entre le calage de l'échelle grossière en multi-échelle et le calage de l'échelle fine en mono-échelle, ne permet pas de conclure sur un quelconque avantage du calage multi-échelle sur le calage mono-échelle. Cette étude a soulevé aussi de nombreuses questions qu'il reste à étudier, telles le passage à une grille 3D, l'étude d'un cas plus réaliste.

Ensuite nous avons testé une adaptation de la synthèse de texture aux problématiques du réservoir, avec l'algorithme de simulation multipoint par patch. Cette méthode est fondée sur l'utilisation d'une image d'entrainement pour construire des images de structures complexes comme les méandres, les chenaux, les fractures, *etc*. L'image d'entrainement représente les structures que l'on attend dans le réservoir. Nous avons tout d'abord testé la sensibilité de l'algorithme aux différents paramètres, avant de proposer une nouvelle méthode d'intégration des données qui combine les avantages du chemin régulier avec l'obligation de trouver un motif qui s'ajuste aux données dures. Nous avons aussi combiné cet algorithme avec la méthode de déformation graduelle, pour se donner la possibilité de l'inclure dans un calage d'historique de production par exemple. Enfin nous avons implémenté deux techniques d'accélération de l'algorithme, qui sont le scan partiel de l'image d'entrainement et la classification par k-means de la base de données. Ces techniques permettent effectivement de diviser jusqu'à 5 fois le temps de calcul mais ne préservent pas la qualité de l'image finale. Le compromis pour garder

une image finale ressemblante est de scanner une assez grande partie de l'image ou d'introduire le concept de cohérence dans la classification. Cet algorithme montre des résultats assez bons mais des temps de calcul encore trop importants, et ne gère pas au mieux certaines images contenant de grandes structures.

Finalement, nous proposons une version multi-échelles de l'algorithme précédent, réduite à deux échelles par souci de simplification. Il faut deux images d'entrainement de résolution celle des échelles, *ie* il faut une image d'entrainement de résolution élevée pour la réalisation à l'échelle fine et une image d'entrainement de résolution plus basse pour la réalisation à l'échelle grossière. Dans une première étape on simule la réalisation grossière à partir de l'image d'entrainement grossière avec l'algorithme mono-échelle. Cette simulation est rapide, car l'image d'entrainement et la grille de simulation sont plus petites. Surtout elle permet de capturer les grandes hétérogénéités avec un template plus petit. Ensuite, il faut créer une base de données un peu plus complexe. Elle doit lister les motifs extraits de l'image d'entrainement fine, et les faire correspondre aux motifs extraits de l'image d'entrainement grossière, pour avoir des motifs double (fin et grossier) qui représente la même chose mais à deux résolutions différentes. La dernière étape est la simulation de l'échelle fine à partir des deux images d'entrainement, conditionnée par la réalisation à l'échelle grossière que l'on vient de simuler. Cette étape est plus rapide que dans le cas mono-échelle car le double template utilisé est plus petit que le template mono-échelle. En effet les grandes structures ont été capturées à l'échelle grossière, il suffit donc de rajouter les détails.

L'étape clé de cette méthode est donc de capturer les grandes lignes de l'architecture de l'image pendant la simulation à l'échelle grossière. Nous avons donc testé différentes manières d'obtenir une image d'entrainement grossière à partir de l'image d'entrainement fine, sur des variables continues et sur des variables discrètes, et leur impact sur la réalisation finale. Pour des variables continues on a montré que dans le cas où l'image était peu contrastée on pouvait rencontrer quelques problèmes pour simuler de bonnes images. Une des solutions proposée est de filtrer l'image pour extraire les grandes structures à l'échelle grossière. Pour des variables binaires, ici des fractures, nous avons obtenu des résultas contraires à notre intuition première. En effet, l'image d'apprentissage grossière obtenue par la médiane, ne ressemble pas du tout à l'image d'apprentissage fine et pourtant c'est elle qui donne les meilleurs résultats.

Pour finir, même si cette méthode est plus rapide que l'approche mono-échelle, on a testé les deux techniques d'accélération précédemment utilisées. Ces techniques ont démontré qu'elles permettaient de diviser le temps de calcul par 5 mais que dans le cas mono-échelle qu'elles dégradaient beaucoup trop l'image finale. Ici, la réalisation fine étant contrainte par la réalisation grossière, l'impact de la réduction du nombre de motifs scannés est moindre. Même avec seulement un quart de l'image scannée, on obtient une réalisation finale tout à fait acceptable.

Perspectives

L'objectif principal de cette approche multi-échelles est bien évidemment d'améliorer le processus du calage. Nous espérons que l'introduction d'une échelle grossière donnera plus de flexibilité au calage, le rendra plus efficace car il y aura moins de paramètres à modifier. Nous ne pourrons vraiment commencer à répondre à cette question que lorsque les méthodes décrites ci-dessus seront consolidées et appliquées à des cas tests plus réalistes. Et il reste encore à travailler l'intégration des données dures et molles dans le cas multi-échelles, ainsi que le passage à la 3D.

Nous avons vu le développement de modèles hiérarchiques qui mixaient les différentes techniques de simulation. Il serait intéressant de voir les résultats obtenus en simulant l'étape à l'échelle grossière avec par exemple de la synthèse de texture et l'étape à l'échelle fine avec du SGSim. Rappelons que SGSim apporte plus de variabilité dans les résultats que la synthèse de texture (point qui lui est souvent reproché).
A partir d'une image binaire de chenaux, on simulerait à l'échelle grossière l'emplacement des chenaux par synthèse de texture. Cette simulation est très rapide puisque l'image d'entrainement a une faible résolution. Ensuite la deuxième étape demande quelques tests mais l'idée serait d'utiliser l'information à l'échelle grossière pour contraindre la réalisation à l'échelle fine en simulant avec SGSim multi-échelles.

Références

Aanonsen, S.I., Eydinov, D., 2006, A multiscale method for distributed parameter estimation with application to reservoir history matching, Comput. Geosci., 10, 97-117.

Aradhye, H.B, Bakshi, B.R, Strauss, R.A, Davis, J.F., 2003, Multiscale SPC using wavelets: theoretical analysis and properties. Process system engineering, 49(4), 939-958

Arpat, G.B., Caers, J., 2005, A multiple-scale, pattern-based approach to sequential simulation, Geostatistics Banff 2004, Quantitative Geology and Geostatistics, 14(1), 255-264

Arpat, G.B., Caers, J., 2007, Conditional simulation with patterns, Mathematical Geology, 39(2), 177-203

Ashikhmin, M., 2001, Synthesizing Natural Textures, The proceedings of 2001 ACM Symposium on Interactive 3D Graphics, Research Triangle Park, NorthCarolina March 19-21, 217-226

Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., Werman, M., 2001, Texture mixing and texture movie synthesis using statistical learning, IEEE Transactions on Visualization and Computer Graphics, 7, 2, 120-135

Behrens, R., Tran, T., 1998, Incorporating seismic data of intermediate vertical resolution into 3D reservoir models. In proceedings of the SPE 49143: Society of Petroleum Engineers Annual Technical conference and Exhibition, New Orleans, LA, USA

Bissell, R.C., Sharma, Y., Killough, J.E., 1994, History Matching using method of gradients: two cases studies, SPE Annual Technical Conference and Exhibition.

Blinn, J.F., Newell, M.E., 1976, Texture and reflection in computer generated images, Communication of the ACM 19, 10 (December), 542-547.

Bouzarkouna, Z., 2012, Well placement optimization, PhD thesis, Université Paris Sud 11

Bridge, J.S., Leeder, M.R., 1979, A simulation model of alluvial stratigraphy: Sedimentology, 26, 617-644

Buades, A., 2005, a non-local algorithm for image denoising, Computer Vision and Pattern Recognition, 2, 60-65, doi: 10.1109/CVPR.2005.38

Burt, P.J., Adelson, E.H., 1983, The Laplacian pyramid as a compact image code, IEEE Transaction on Communications, 31, 4, 532-540

Caers, J., 2003, Geostatistical history matching under training-image based geological constraints, SPE J., 8(3), 218-226

Carrera, J., Neuman, S.P., 1986, Estimation of aquifer parameters under transient and steady state conditions, Water Resources Research, 22(2).

Catmull, E.E., 1974, A Subdivision Algorithm for Computer Display of Curved Surfaces, PhD Thesis, Computer Science Department, University of Utah, Salt Lake City.

Cellis, M.R., Dennis, J.E., Tapia, R.A., 1985, A trust region strategy for non-linear equality constrained optimization, Numerical Optimization, 71-82

Chatterjee, S., Dimitrakopoulos, R., Mustapha, H., 2012, Dimensional reduction of pattern-based simulation using wavelet analysis, Mathematical Geosciences, 44, 343-374

Conn, A., Scheinberg, K., Vicente, L., 2009, Introduction to derivative free optimization, MPS-SIAM Series on optimization.

Corey, A.T., 1977, Mechanics of heterogeneous fluids in porous media, Water Resources publications, Fort Collins, Colorado.

Crouse, M.S., Nowak, R.D., Baraniuk, R.G., 1998, Wavelet-based statistical signal processing using hidden Markov models, IEEE Trans Signal Process, 46(4), 886-902

De Bonet, J., 1997, Multiresolution Sampling procedure for analysis and synthesis of texture images, In SIGGRAPH97, pp361-368

De Marsily, G.,Delay, F., Gonçalvès,J., Renard, P., Teles, V., Violette, S., 2005, Dealing with spatial heterogeneity, Hydrogeology Journal, 13(1), 161-183

De Montleau, P., Cominelli, A., Neylon, D.R.K., Pallester, I., Tesaker, O., Nygoud, I., 2006, Production optimization under constraints using adjoint gradients, Proceedings of 10th ECMOR

Desbarats, A.J., 1996, Modeling spatial variability using geostatistical simulation, In: Rouahni, S., Srivastava, R.M., Desbarats, A.J., Cromer, M.V., Johnson, A.I., Geostatistic for Environmental and Geotechnical Applications, American Society for Testing and Materials STP 1283, Philadelphia, pp32-48.

Deutsch, C.V., Lewis, R., 1992, Advances in the practical implementation of indicator geostatistics, In: proceedings of the 23rd International APCOM Symposium, Tucson, AZ, Society of Mining Engineers, pp 169-179

Deutsch, C.V., Wang, L., 1996, Hierarchical Object-Based Stochastic Modeling of Fluvial Reservoirs, Mathematical Geology, 28(7), 857-880

Deutsch, C. V., and Journel, A.G., 1998, GSLIB: Geostatistical software library and user's guide, 2nd ed.: Oxford University Press, New York, 369 p.

Droujinine, A., 2006, Multiscale geophysical data analysis using the Eigen image discrete wavelet transform. Journal of Geophysics and Engineering, 3, 59-69

Duda, R.O., Hart, P.E., Stork, D.G., 2001, Pattern classification, edited by Wiley-interscience

Duranleau, F., 2008, Génération et édition de textures géométriques représentées par des ensembles de points, Thèse de doctorat, Département d'informatique et de recherche opérationnelle, Université de Montréal.

Efendiev,Y., Hou, T.Y., Luo, W., 2006, Preconditioning Markov chain Monte Carlo simulations using coarse-scale models, SIAM J. Sci. Comp., 28, 776-803

Efros, A.A., Leung, T.K., 1999, Texture synthesis by non-parametric sampling, IEEE International Conference on Computer Vision, Corfu, Greece

Efros, A.A., Freeman, W.T., 2001, Image quilting for texture synthesis and transfer, paper presented at Proceedings of ACM SIGGRAPH Conference on computer graphics.

El Ouassini, A., Saucier, A., Marcotte, D., Favis, B., 2008, A patchwork approach to stochastic simulation: A route towards the analysis of morphology in multiphase systems, Chaos Solitons and Fractal, 36(2), 418-436

Evensen, G. 1994. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics, J. Geophys. Res., 99, 10,143-10,162

Evensen, G., 2009, Data assimilation, the Ensemble Kalman Filter, 2nd Edition, Springer.

Fenwick, D., Thiele, M., Agil, M., Hussain, A.P., Humam, F., Caers, J., 2005, Reconciling prior geologic information with production data using streamlines: application to a giant Middle-Eastern oil field, SPE ATCE, Dallas, TX, USA, SPE 95940.

Froidevaux, R., 1992, Probability field simulation, Geostatistics Troia'92, proceedings of the Fourth Geostatistics Congress, A. Soares (ed.), Kluwer Academic Publishers, Dordrecht, Netherland.

Gao, G., Li, G., Reynolds, A.C., 2007, A stochastic optimization algorithm for automatic history matching, SPE Journal, 12(2), 196-208

Garcia, M., Froidevaux, R., 1997, Application of geostatistics to 3D modelling of contaminated sites: a case study, In: Soares, A., Gomez-Hernandez, J., Froidevaux, R., geoENV I – Geostatistics for Environmental Applications, Kluwer Academic publishers, Dordrecht, pp309-325.

Gardet, C., Le Ravalec, M., 2014, Multiscale Multiple Point Simulation Based on Texture Synthesis, 14th European Conference on the Mathematics of Oil Recovery Catania, Sicily, Italy, 8-11 September

Gardet, C., Le Ravalec, M., Gloaguen, E., 2014, Multiscale Parameterization of Petrophysical Properties for Efficient History-Matching, Mathematical Geosciences, 46(3), 315-336, doi: 10.1007/s11004-013-9480-3

Gervais, V., Roggero, F., 2010, Integration of saturation data in an history matching process based on adaptive local parameterization, J. Pet. Sci. Eng., 73(1-2), 86-98.

Gloaguen, E., Dimitrakopoulos, R., 2009, Two dimensional conditional simulation based on wavelet decomposition of training image, Mathematical Geosciences, 41(7), 679-701

Gomez-Hernandez, JJ., Sahuquillo, A., Capilla, J.E., 1997, Stochastic simulation of transmissivity fields conditional to both transmissivity and piezometric data - 1.Theory, J. Hydrol., 203(1-4), 162-174.

Goovaerts, P., 1997, Geostatistics for Natural Resources Evaluation, Oxford University Press, New York, 512 p.

Gu, Y., Oliver, D.S., 2005. History matching of the PUNQ-S3 reservoir model using the Ensemble Kalman filter, SPE J., 10, 51-65.

Guardiano, F., Srivastava, R.M., 1993, Multivariate geostatistics: Beyond bivariate moments, in Soares, A., ed., Geostatistics-Troia, Vol.1: Kluwer Academic, Dordrecht, 133-144

Gutjahr, A., Wilson, J., 1989, Co-kriging for stochastic flow models: Transport in Porous Media, 4(6), 585-598.

Haldorsen, H.H., Damsleth, E., 1990, Stochastic modeling, J. Pet. Technol., 42, April, 404-412

Han, C., Risser, E., Ramamoorthi, R., Grinspun, E., 2008, Multiscale texture synthesis, Jouranl ACM Transactions on graphics (TOG), Proceedings of ACM SIGGRAPH'08, 27(3)

Hanwook, P., Haewon, B., Changhun, K., 2013, Multi-exemplar inhomogeneous texture synthesis, Computer and Graphics, 37, 54-64.

Heeger, D.J., Bergen, J.R., 1995, Pyramid-based texture analysis/synthesis, In SIGGRAPH95, pp229-238

Hertzman, A., Jacobs, C.E., Olivier, N., Curless, B., Salesin, D.H., 2001, Image analogies, SIGGRAPH 2001, 327-340

Hewett, T.A., 1986, Fractal Distributions of Reservoir Heterogeneity and their Influence on Fluid Transport, Conference Paper, SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana

Honarkhah, M., Caers, J., 2010, Stochastic simulation of patterns using distance-based pattern modeling, Mathematical Geosciences, 42, 487-517

Hu, L.Y., 2000, Gradual deformation and iterative calibration of Gaussian-related stochastic models, Math. Geol., 32(1), 87-108.

Hu, L., Chugunova, T., 2008, Multiple-point geostatistics for modeling subsurface heterogeneity: A comprehensive review, Water Resources Research, 44, W11413

Jafari, A., Babadagli, T., 2010, Prediction of the equivalent fracture network permeability using multivariable regression analysis and artificial neural network, Proceedings of World Geothermal Congress, Bali, Indonesia, 25-29 April

Jafarpour, B., McLaughlin, D.B., 2007, Efficient permeability parameterization with the discrete cosine transform, SPE RSS, Houston, TX, USA, SPE 106453.

Journel, A.G., Gundeso, R., Gringarten, E., Yao, T., 1998. Stochastic modeling of a fluvial reservoir: a comparative review of algorithms, J. Petro. Sci. Eng., 21, 95–121.

Journel, A.G., 2005, Beyond covariance: the advent of multipoint geostatistics, Geostatistics Banff2004, Quantitative Geology and Geostatistics, 14, 225-233

Kolbjørnsen, O., Stien, M., Kjønsberg, H., Fjellvoll, B., Abrahamsen, P., 2014, Using multiple grids in Markov mesh facies modeling, Mathematical Geosciences, 46, 205-225

Koltermann, C., Gorelick, S., 1996, Heterogeneity in sedimentary deposits: A review of structure-imitating, process-imitating, and descriptive approaches, Water Resources Research, 32(9), 2617-2658

Lee, S.H., Malallah, A., Datta-Gupta, A., Higdon, D., 2000, Multiscale data integration using Markov Random Fields. In proceeding of SPE 63006: Society of Petroleum Engineers Annual Technical conference and Exhibition, Dallas, TX, USA

Le Ravalec, M., 2010, Pilot block method methodology to calibrate stochastic permeability fields to dynamic data, Mathematical Geosciences, 42, 165-185

Le Ravalec-Dupin, M., Da Veiga, S., 2011, Cosimulation as a perturbation method for calibrating porosity and permeability fields to dynamic data, Computers & Geosciences, 37(9), 1400-1412.

Le Ravalec-Dupin, M., Da Veiga, S., Derfoul, R., Enchéry, G., Gervais, V., Roggero, F., 2012, Integrating Data of Different Types and Different Supports into Reservoir Models, Oil Gas Sci. Technol. – Rev. IFP Energies Nouvelles, 67 (5), 823-839, doi: 10.2516/ogst/2012024

Le Ravalec, M., Doligez, B., Lerat, O., 2014, Integrated reservoir characterization and modeling, ebook, doi: 10.2516/ifpen/2014001

Li, H., Caers, J., 2011, Geological modelling and history matching of multi-scale flow barriers in channelized reservoirs: methodology and application, Petroleum Geoscience, 17, 17-34

MacQueen, J.B., 1967, Some methods for classification and analysis of multivariate observations, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1, University of California Press, 281–297

Mahmud, K., Mariethoz, G., Caers, J., Tahmasebi, P., Baker, A., 2014, Simulation of Earth textures by conditional image quilting, Accepted Article, Water Resources Research, doi: 10.1002/2013WR015069

Mariethoz, G., Renard, P., Straubhaar, J., 2010, The direct sampling method to perform multiple-point geostatistical simulations, Water Resources Research, 46,W11536

Mariethoz, G., Lefebvre, S., 2014, Bridges between multiple-point geostatistics and texture synthesis: Review and guidelines for future search, Computers & Geosciences, http://dx.doi.org/10.1016/j.cageo.2014.01.001

Marsily, G., de Lavedan, G., Boucher, M., Fasanino, G., 1984, Interpretation of interference tests in a well field using geostatistical techniques to fit the permeability distribution in a reservoir model, in Geostatistics for Natural Resources Characterization, edited by Verly, G., Michel, D., Journel, A.G., and Marechal, A., NATO ASI Series C 182, 831-849, Reidel, D., Norwell, Mass

Nelder, J.A., Mead, R., 1965, A simplex method for function minimization, Computer Journal, 7, 308-313

Omre, H., 1991, Stochastic model for reservoir characterization, in Kleppe, J., Skjaeveland, S.M., eds., Recent advances in improved oil recovery methods for North Sea sandstone reservoirs: Norwegian Petroleum Directorate, Stavanger, 14p.

Parra, A., Ortiz, J.M., 2011, Adapting a texture synthesis algorithm for conditional multiple point geostatistical simulation, Stochastic Environmental Research and Risk Assessment, 25, 1101-1111

Portilla, J., Simoncelli, E.P., 2000, A paramedic texture model based on joint statistics of complex wavelet coefficients, International Journal of Computer Vision, 40, 49-71

Popat, K., Picard, R., 1993, Novel cluster-based probability model for texture synthesis, classification and compression, Visual Communication and Image Processing, 756-768

RamaRao, B.S., Lavenue, A.M., Marsily, G., de Marietta, M.G., 1995, Pilot point methodology for automated calibration of an ensemble of conditionally simulated transmissivity fields, Part 1, Theory and computational experiments, Water Resour. Res., 31(3), 475-493.

Renard, P., Straubhaar, J., Caers, J., Mariethoz, G., 2011, Conditionning facies simulations with connectivity data, Math Geosci 43(8), pp. 879-903

Rezaee, H., Mariethoz, G., Koneshloo, M., Asghari, O., 2013, Multiple-point geostatistical simulation using the bunch-pasting direct sampling method, Computers & Geosciences, 54, pp 293-308

Romary, T., 2010, Integrating production data under uncertainty by parallel interacting Markov chains on a reduced dimensional space, Comput. Geosci., 13(1), 103-122.

Sahni, I., Horne, R.N., 2005, Multiresolution wavelet analysis for improved reservoir description, SPEREE, 8(1), 53-69.

Shannon, C. E., 1948, A mathematical theory of communication, Bell Sys. Tech. Journal, 27

Straubhaar, J., Renard, P., Mariethoz, G., Froidevaux, R., Besson, O., 2011, An improved parallel multiple-point algorithm using a list approach, Mathematical Geosciences, 16(3), 779-797

Straubhaar, J., Malinverni, D., 2014, Addressing conditioning data in multiple-point satistics simulation algorithms based on multiple grid approach, Mathematical Geosciences, 46, 187-204

Strebelle, S., 2000, Sequential simulation drawing structures from training images, PhD. thesis, Stanford University, Stanford, California

Strebelle, S., 2002, Conditional Simulation of Complex Geological Structures Using Multiple-Point Statistics, Mathematical Geology, 34 (1), 1-21

Strebelle S., Zhang, T., 2005, Non-stationary multiple-point geostatiscal models. In : Leuangthong O, Deutsch C (eds) Geostatistics Banff 2004, vol1, Springer, Dordrecht, pp. 235-244

Sun, N.-Z., 1995. Inverse Problems in Groundwater Modeling, Kluwer Academic Publishers, Dordrecht, The Netherlands, 337 pp.

Tahmasebi, P., Hezarkhani, A., Sahimi, M., 2012, Multiple-point geostatistical modelling based on the cross-correlation functions, Computational Geosciences, 16, 779-797

Tahmasebi, P., Sahimi, M., Caers, J., 2014, MC-CCSIM: accelerating pattern-based geostatistical simulation of categorical variables using a multi-scale search in Fourier space, Computers & Geosciences, http://dx.doi.org/10.1016/j.cageo.2014.03.009

Tran, T.T., Wen, X.H., Behrens, R.A., 1999, Efficient conditioning of 3D fine scale reservoir model to multiphase production data using streamline-based coarse-scale inversion and geostatistical downscaling, SPE ATCE, Houston, TX, USA, SPE 56518.

Tran, T., Mueller, U.A., Bloom, L.M., 2002, Multiscale conditional simulation of two-dimensional random processes using Haar wavelets, In Proceedings of Geostatistical Association of Australasia Symposium, Perth, 56-78

Trangmar, B.B., Yost, R.S., Uehara, G., 1985, Application of geostatistics to spatial studies of soil properties, Advances in Agronomy, 38, pp 45-94

Wei, L., Levoy, M., 2000, Fast texture synthesis using tree-structured vector quantization, In Proceedings of SIGGRAPH 2000

Wei, L., Lefebvre, S., Kwatra, V., Turk, G., 2009, State of the Art in Example-based Texture Synthesis, EUROGRAPHICS 2009/M. Pauly and G. Greiner

Wu, J., Boucher, A., Zhang, T., 2008, A SGeMS code for pattern simulation of continuous and categorical variables: FILTERSIM, Computers & Geosciences, 34 (12), 1863-1876

Yeh, W.W.G., 1986. Review of parameter identification procedures in groundwater hydrology: The inverse problem. Water Resour. Res., 22, 95–108.

Zelinka, S., Garland, M., 2004, Jump map-based interactive texture synthesis, ACM Trans. Graph., 23(4), 930-962

Zhang, T., Switzer, P., Journel, A. G., 2006, Filter-based classification of training image patterns for spatial simulation, Mathematical Geology, 38(1), 63-80