Centre d'hydrogéologie et de géothermie
Université de Neuchâtel

Master en Hydrogéologie et Géothermie

3D Multiple-point Statistics Simulations of the Roussillon Continental Pliocene Aquifer using DeeSse

Valentin Dall'Alba

Supervised by Prof. Philippe Renard
and Yvan Caballero

February 22, 2019

Abstract

This study presents a novel workflow that was developed to model the internal heterogeneity of a complex 3D aquifer using the Multiple-point Statistics (MPS) algorithm DeeSse. The modelled aquifer is the Continental Pliocene layer (PC) that is part of the Roussillon Aquifer in the Perpignan's region in Southern France.

This work is part of the Dem'Eaux Roussillon research project that aims to characterize the whole groundwater dynamics of the Roussillon Aquifer in a context of growing population, climate change, and increasing pressure on the freshwater resources in a Mediterranean environment. The specific purpose of the Master project is to model the 3D heterogeneity of the geology of the PC aquifer. The results of this work will then be used in further steps to characterize the sustainability of groundwater flow extraction and in particular the risks of seawater intrusion in the plain that may affect the groundwater quality.

For this purpose, we use the direct sampling algorithm DeeSse and demonstrate its applicability for the first time on a large study site. New procedures are proposed to account for known geological constraints during simulations. The interpretation of gamma ray and resistivity logs in terms of facies provided the hard data used to constrain the geostatistical simulations along the boreholes. Then, to represent the complex sedimentation history of the plain, a non-stationary training image (TI) is used. In order to control where each type of geological environment may occur in the plain, novel procedures are proposed to generate auxiliary variables maps by solving numerically a flow problem enabling to obtain plausible trends between the sources of the sediments and the outlet of the sedimentary system. These trend and rotation maps are based on geological insights gathered from outcrops and a general knowledge of processes occurring in these types of sedimentary environments. Several sets of 100 simulations are produced and analyzed statistically. Facies probability maps and vertical proportion curves are analysed to test the plausibility of the model. As compared to previous studies using MPS, this is the first time that such a multivariate approach is employed at a regional scale.

Finally, we demonstrate that the hydraulic conductivity of the different sedimentary deposits cannot be represented by unique values if we assume that the geological models that we have generated are acceptable. Based on transmissivity estimation, we conclude that the hydrological characteristics of the deposits are heterogeneous over the PC aquifer.

Résumé

Ce travail de Master présente les nouvelles méthodes développées pour modéliser l'hétérogénéité interne d'un aquifère 3D complexe à l'aide de l'algorithme Statistiques Multi-points (MPS) DeeSse. L'aquifère modélisé est la couche Pliocène Continentale (PC) qui fait partie de l'aquifère du Roussillon, dans la région de Perpignan dans le Sud de la France.

Cette étude s'inscrit dans le projet de recherche Dem'Eaux Roussillon qui vise à caractériser l'ensemble des processus hydrodynamiques souterrains de l'aquifère du Roussillon dans un contexte d'augmentation de la population, de changements climatiques et de l'augmentation de la pression sur la ressource en eau dans un environnement méditerranéen. Le but principal est de modéliser l'hétérogénéité géologique 3D de l'aquifer PC. Les résultats de ce travail de recherche seront ensuite utilisés dans les prochaines étapes de caractérisation de la ressource en eau souterraine expoitable et de la caractérisation du risque d'intrusion d'eau salée dans la plaine, qui pourrait affecter la qualité de l'eau.

C'est dans ce but que nous utilisons l'algorithme d'échantillonnage direct DeeSse et démontrons son applicabilité pour la première fois sur un grand site d'étude. De nouvelles procédures sont proposées pour tenir compte des contraintres géologiques connues durant les simulations. L'interprétation de diagraphies gamma-ray et résistivité en terme de faciès fournit les données conditionnantes utilisées pour contraindre les simualtions géostatistiques le long des forages. Par la suite, afin de représenter l'histoire complexe de sédimentation dans la plaine, une image d'entrainement (TI) non-stationnaire est utilisée. Pour contrôler la distribution spatiale des environnements de déposition, de nouvelles procédures de création de variables secondaires basées sur la résolution numérique d'équations d'écoulement sont ici proposées. Celles-ci permettent de prendre en compte les sources d'apports de sédiments, les exutoires du système et ainsi de créer des cartes de tendances de sédimentation réalistes. Les cartes de tendance et de rotation sont crées à partir des informations provenant des affleurements et des campagnes de travaux de terrain ainsi que des connaissances générales des processus liés à ce type d'environnement sédimentaire. Plusieurs sets de 100 simulations ont ainsi été réalisés et analysés statistiquement. Des cartes de probabilités d'occurrence des faciès ainsi que des courbes de proportion verticale ont également été produites et analysées pour tester la robustesse du modèle. En comparaison avec les précèdentes études utilisant les techniques MPS, c'est ici la première fois qu'une telle approche multi-variables est utilisée à si grande échelle.

Finalement, nous démontrons que la conductivité hydraulique des différents faciès sédimentaires ne peut être décrite par des valeurs uniques, si nous assumons que les modèles générés sont acceptables. En se basant sur une estimation de la transmissivité, nous concluons que les paramètres hydrologiques des dépots sont hétérogènes pour l'aquifère du Pliocène Continental.

Remerciements

Je tiens tout d'abord à remercier le Pr. Philippe Renard qui m'a proposé ce projet et qui m'a inspiré tout au long de ce travail de Master par son dynamisme intellectuel et sa motivation à toute épreuve.

Je tiens également à remercier le BRGM, partenaire de ce projet, et tout particulièrement Yvan Caballero pour le temps accordé au suivi de mon projet, à la mise en place des journées de terrain et des réunions avec les différents acteurs du projet.

Merci également à Benoit Issautier, Eric Lasseur et Cédric Duvail pour leur support et expertise. Je suis reconnaissant pour le temps accordé lors des journées de terrain et réuinons, j'ai beaucoup appris de nos discussions.

Un grand merci également à Julien Straubhaar pour ses nombreux conseils et coups de main pendant le projet, ainsi que pour l'aide apportée pour les simulations.

Merci également à tous les professeurs, assistants, doctorants et personnels du CHYN pour ces deux années de Master.

Je n'oublie pas mes camarades de Master avec qui j'ai partagé de nombreux bons moments durant ces deux années.

Je remercie infiniment toute ma famille pour le soutien appporté le long de mes études, leur bienveillance et leurs encouragements continus.

Enfin, je remercie personnellement Charlène qui m'a beaucoup aidé pour terminer ce manuscrit et avec qui j'ai passé deux années formidables ici à Neuchâtel.

Ce projet de Master m'a permis de developper de nombreuses compétences tout en éveillant ma curiosité pour la recherche. Je ne peux être que reconnaissant d'avoir travaillé sur ce projet et je remercie encore une fois toutes les personnes impliquées.

Contents

1 Introduction

The aim of this project was to model the geological facies of the Continental Pliocene (PC) aquifer of the Roussillon Plain. The aquifer is situated in the South-West of France in the Perpignan region. The project was done in close collaboration with the French Geological Survey (BRGM) and is part of the Dem'Eaux Roussillon project. The PC layer is composed of alluvial deposits and presents a high level of heterogeneity. Those types of deposits are generally hard to understand regarding their spatial distribution and even harder to model due to their high heterogeneous nature. From its social and economic importance, a clear understanding of the aquifer is essential to the authorities for a long term and sustainable management of the regional water resources. The characterization of the geology and the hydrogeological properties of the aquifer and its interactions that occurred at the aquifer's limits are necessary to understand the hydrodynamic system.

Geostatistical methods have been used in the last decades to model the heterogeneity of the sub-surface. These methods are widely used in different fields going from risks assessment, resources management, mining or in the petroleum industry for numerical reservoir modelling [Matheron 1963, Strebelle et al. 2002, de Carvalho et al. 2017]. The geostatistical approach offers different sets of tools, their aims are to infer variables of interest at locations where they have not been measured. Those algorithms use hard data to constrain their simulations. Hard data are information or measurements gathered on the study site, these data are geo-referenced and are used to interpolate the missing parts of the simulation.

One well-known and largely used geostatistical method is the kriging method [Matheron 1963]. This algorithm infers information by calculating the best linear unbiased estimator between unknown points and hard data. Kriging estimation is fast, simple and produces smooth interpolation. However the kriging interpolation tends to decrease the variability of the data, cannot reproduce known geological object features (for example the sinuosity of a channel) and thus is not always well adapted to estimate variables for geological reservoirs. The kriging algorithm has its strengths and weaknesses and is one of many ways to spatially interpolate data.

Multiple-point Statistics (MPS) is another geostatistical approach. While the solution of kriging is unique by definition, MPS is a stochastic method that generates a set of equiprobable solutions [Renard et al. 2013]. Stochastic models consider the parameters as random variables or distributions, whereas deterministic models consider these parameters to be perfectly known and defined by a unique value. The outputs of stochastic simulations are a set of equally likely solutions rather than one unique solution. As nature produces het-

erogeneous systems and regarding the small amount of measurement generally available, the stochastic approach is an efficient and powerful tool to simulate the sub-surface heterogeneity and especially to quantify the uncertainties of the model [Renard et al. 2013]. Moreover, MPS algorithms allow to integrate a conceptual knowledge of the variable of interest into the simulation with the use of a training image (TI), which represents a conceptual model of the variables aimed to be simulated.

Different MPS algorithms have been developed over the years: SNESIM [Strebelle et al. 2002], IMPALA [Straubhaar et al. 2011], FILTERSIM [Zhang et al. 2006] and Direct Sampling [Mariethoz et al. 2010]. These algorithms use different approaches to store the data, to sample the TI and can work with different variable types. We selected the Direct Sampling algorithm, implemented in the softawre DeeSse, to perform this study. DeeSse is a the Multiple-point Statistics (MPS) code [Straubhaar 2017], developed at the Center of Hydrogeology and Geothermics of University of Neuchâtel (CHYN). This code allows to work with a multivariate approach and to use complex continuous rotation and affinity maps as auxiliary variables.

In this Master project, the aim being to model the geological facies of the Continental Pliocene (PC) aquifer of the Roussillon Plain, we propose to use the multivariate approach provided by DeeSse to construct realistic geometric patterns constrained by 3D trend maps and lithology logs analysis. This is the first time that this tool is applied for modelling a large and complex aquifer. To reach that objective, it was necessary to develop new procedures and a workflow to include most of the existing geological knowledge into the stochastic method.

The overall procedure contains several steps. It was decided first to work with a 2D TI, and to create the 3D model by stacking 2D simulations with a proper 3D evolution of the trends. It appeared that the creation of a 3D TI would have been too complex to create and to constrain. We decided to work in a 3D grid with a rather fine resolution along the z axis ($2m$), which corresponded to the minimal vertical dimension of the facies found in the plain. This allowed us to by-pass the creation of a 3D TI and to simulate in 2D with object of a realistic z dimension.

We then created and tested several non stationnary 2D TI that displayed the main features of the entire sedimentation process of the plain. In order to constrain the TI, we created 2D trends maps. We used a flow and transport simulation approach to simulate the horizontal trend of the plain. By using a groundwater flow simulator, we were able to produce complex horizontal trends corresponding to the sedimentation of the plain. This approach was also applied to create the vertical trend (progradation of the plain towards the sea). By simulating different representative 2D trend layers and combining them together through

the vertical axis, we were finally able to create a complex 3D trend map, which accounted for both vertical and lateral sedimentation trends that occur on the plain. This approach is unique and has never been proposed for the creation of MPS auxiliary variables. Moreover, the use of the direct sampling algorithm allows us to work with complex rotation maps, where the rotation value is continuous through all the nodes of the grid [Mariethoz et al. 2010], whereas classic MPS techniques require to define rotation zones of unique value [de Carvalho et al. 2017]. We created two continuous rotation maps that were used as rotation bounds for the simulation.

The final model was finally tested by generating several sets of 100 simulations. From them, we calculated probability maps, a Shannon Entropy map and vertical proportion curves from the final model, to characterize its uncertainties. We also tested two methods to estimate the hydraulic conductivity of the simulated facies from measured transmissivity values. The multivariate approach, which combined a non-stationnary TI and a complex 3D trend map, has succeeded in the production of a large and realistic 3D aquifer. The complex structure of the plain has been well reproduced through the simulations and the realistic aspect of the facies has been honored. However, from post-simulation analysis it appeared that some constraints applied on the simulation were too strong. A small variability was observed within the post-simulated maps in the central part of the plain. Additionally the characterization of the hydraulic parameters could not be determined with a simple transmissivity inversion method. It is likely that those parameters are heterogeneous over the plain and thus cannot be described by unique values.

This manuscript is structured as follows. Section 2 introduces some background information regarding the project, the geology and hydrogeology of the Roussillon Aquifer, followed by an introduction on geostatistics and the DeeSse algorithm. The understanding of geological settings is essential to produce well defined inputs for the simulations. Section 3 describes the overall workflow and all the implementation details: creation of grids, borehole log interpretation, training image (TI) and auxiliary maps creation, as well as the simulation parameters. Finally in section 4, we present and discuss the results of the simulations and the post-simulations calculations.

2 State of the art

In this section, I review some general information regarding firstly the geology and hydrogeology of the Roussillon Plain and secondly on the Multiple-point Statistics approach and the DeeSse algorithm. In order to produce realistic simulations, we need to understand what the sedimentary processes are before being able to create a suitable training image for the simulations. The same principle stands for the algorithm, we need to understand how it works and what are the main parameters controlling the simulation.

2.1 Description of the Roussillon Plain

2.1.1 Geology of the Plain

Located on the South-West part of France, between the Oriental Pyrenees Mountains and the Mediterranean Sea, the Roussillon Aquifer is a multi-layers aquifer covering a $900km^2$ area. This aquifer is composed of different sandstone units which are separated by silt and clay layers of low permeability [Duvail 2007]. This basin originates with the opening of the Gulf of Lion (Oligocene to Miocene). Then, this basin has been largely eroded due to the Messinian Salinity Crisis (MSC). It is with the drawback of the Mediterranean Sea level that the Miocene layer has been exposed and eroded [Lofi et al. 2005]. During the Pliocene, this basin has been filled up first with large fluvio-marine sedimentary prisms (Sandy Marine Pliocene or PMS) and then with a continental river system (Continental Pliocene or PC). On top of the stratigraphic pile, Quaternary deposits associated with river and lagoon systems are found. They have cut through the PC deposits and formed the actual landscape of the Plain. The marine structures are composed of large sandy bands, which can be more than one km in extent and over $100m$ thick. Due to the subsidence of the margin, the prisms are now tilted toward the sea. As presented on the figure 1, these prisms also extend under the sea and form the main part of the Roussillon Basin.

Regarding the PC deposits, the main source of sediment came from the weathering of the massifs that surround the Roussillon Plain. We find on the North the Corbière massif composed of limestone, on the West the Canigou massif composed of granite, gneiss and schist rocks and on the South border the Albera massif composed of granitic and metamorphic formations [Dutartre et al. 1995]. Near the relief, the association between high energy systems and the large amount of available sediments started by creating alluvial fan deposits. The particle size of these deposits is large with a sandy matrix. The deposits are unsorted sandstone conglomerates. These alluvial fans have a large extent of 1-3km radius and can

be more than 10m thick. From field observations, there is a large amount of repetition in the deposition that can yield to combined structures of 3-6km in radius and over 60m in thickness. Following this, a network of braided river deposits take place. Sediments that compose those systems are coarse sands and sandstone conglomerates. There is still a large amount of energy in the system which results in unsorted clasts. These braided structures have generally an extent of 100-150m width and are 1-5m thick. It appears from field observations, that these networks can be well connected in their lateral extents. Near their sources, these braided networks tend to be very dense and wide, while they gradually narrow as they advance on the plain. With the decrease of the sedimentary slope, the structures tend to evolve to meander rivers [Nichols 2009]. These meander structures have the same type of sediment that the braided river deposits, except that the sediments are better sorted. The meander beds are wider than the braided beds. Their width can go up to 300m and their thickness up to 10m.

The connectivity between the different structures is hard to observe either in the vertical or in the horizontal directions. Three other sedimentary structures are also intrinsically developed within the evolution of the alluvial plain. The two first ones are the floodplain and the crevasse splay. A large rain event can result in a flood in the plain when the amount of water and sediments exceeds the volume capacity of the river [Nichols 2009]. This can also happen if the amount of energy becomes too important, which can lead to the breaking of the river bank. Facies associated with these events, are the floodplain deposits, which are mainly composed of very fine sediments. These deposits are generally a mix between fine sand, clay and silt. Their thickness can be up to 5 m due to the stacking of multiple flood events. The crevasse splay deposits happen when a river bank breaks or when the flow energy is higher than the resistance of the bank. The structure of the crevasse splay is fan-like, with a radius of 100-150m and is generally well sorted. The sediments that compose crevasse splay are very fine with some sparse coarse sand in the matrix. Finally the last structures observable are river levees. These structures are associated with meander beds and are mostly made of sand and silt. These deposits are less visible on the outcrops and thus are not well defined on the plain. At the top of the stratigraphic units, we find Quaternary deposits. They are fluvial deposits in the proximal part of the relief and tend to form lagoon deposits when approaching the sea.

This Master project focuses on the second layer, the Continental Pliocene (Figure 1 - green and orange deposits). From the field observations, 6 facies are aimed to be modelled – alluvial fan deposit, floodplain deposit, braided river deposit, meander river deposit, crevasse splay deposit and levee deposit.

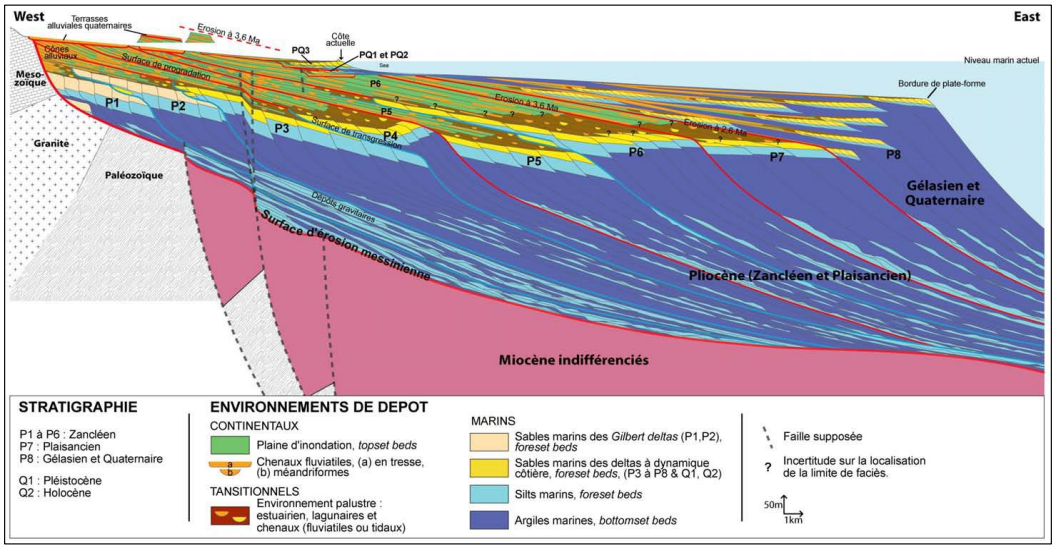


Figure 1 – West-East cross section of the Roussillon Aquifer. The Miocene basement is represented in purple and is covered by the Marine Pliocene prisms in dark and light blue. The Continental Pliocene is represented in green and orange, respectively the flood-plain and the undifferentiated river beds. The Quaternary sits on top of it but is very thin compared to the rest of the deposits. [Duvail 2007]

2.1.2 Hydrogeology of the Plain

The Roussillon Aquifer is essential for the Perpignan region, since it is used for the agriculture, domestic use and controls the sea water intrusion of the Mediterranean Sea. The exploitation of the groundwater resource is very important for this type of region that usually have a large population, a dry and hot climate, an important touristic influx and a local economy based on agriculture. The main reservoirs of the aquifer are composed of the Marine Pliocene with large sandy delta deposits, the Continental Pliocene with sandy braided or meander river deposits and the Quaternary alluvial deposits. The connectivities between these reservoirs are not yet well defined just as the recharge processes of the aquifer.

More than 95% of the population needs in water are provided by the exploitation of the aquifer. With an annual pluviometry of $570 mm/yr$ and a consumption of 80 millions of m^3/yr, the resource vulnerability is important [CLE 2011]. The four main rivers of the plain are (from North to South) – the Agly river, the Têt river, the Canterrane and the Tech river, which take their sources in the nearest massifs that border the plain. Most of the agriculture irrigation is performed through the exploitation of the rivers while the rest is pumped from the aquifer through a network of more than 4500 private wells. Furthermore, due to climate changes, aquifers water volume and their recharge sources could decrease in the future. For a scenario where the average annual temperature increases of 1.5C°and is associated with a decrease in precipitation rate, the rivers flow could drop by 40% over the next 30 years [Chauveau et al. 2013]. The stress on the water resource would automatically increase. Depending on their vertical connectivities, the Quaternary aquifer could be largely impacted by these hydraulic and physical variations, whereas the Pliocene aquifer could be less affected [Caballero and Ladouche 2015]. Finally the Pliocene extent goes under the sea shore and protects the groundwater resource and the agricultural wells from salted-water intrusion. It has been calculated that the sea level increases at a rate of $3 mm/yr$ [Cazenave 2013]. Due to the low altimetry of the plain, a large number of zones could become submerged. Moreover, during large storm events, salted-water intrusions could occur near the shore and pollute extraction wells. The increase of the aquifer use coupled with the increase of the sea level could promote sea-water intrusions and jeopardize the water resource [Dörfliger and Perrin 2012].

The large extent of the Pliocene aquifer, both on the plain and under the sea shore, represent a large water reservoir. However, due to the uncertainties linked to the reservoir geometries, connectivities and recharge processes, the management of this resource appears very difficult. The Dem'Eaux project aims to answer these hydrogeologic issues. All of these questions need to be answered for a sustainable management and for the protection of the resource.

2.2 Geostatistics and Multiple-point Statistics

I now propose to introduce some basic knowledge on the Multiple-point Statistics and the non-stationary nature of the data before presenting the algorithm used for the simulation, DeeSse. I present the different parameters that control the algorithm and the input data that are needed in order to create a simulation.

2.2.1 Multiple-point Statistics Definition

This master project aims to model the Continental Pliocene (PC) aquifer using the advanced Multiple-point Statistics algorithm DeeSse. Multiple-point Statistics (MPS) methods are advanced geostatistical tools used to infer information from prior data. The first MPS algorithms have been developed in the 90's with the ambition to improve the two-points simulation used at that time. It has also been developed with the aim to provide tools and methods that could constrain a simulation not only with a probabilistic approach but also with a geological point of view. This has been made possible with the use of training images (TI). TIs represent a conceptual idea of the structure that is aimed to be simulated [Hu and Chugunova 2008]. Unlike some other geostatistic methods such as semi-variogram, utilization of training image allows specialists from different fields to discuss together about the geometry and the type of heterogeneity of a model. The DeeSse algorithm can also deal with hard data, such as outcrops or borehole data and perform multi-variables simulations [Mariethoz et al. 2010]. The use of TIs give flexibility and creativity to the modeller.

Traditional geostatistics are based on the definition of a random function that should describe the distribution of the modelled object. However, defining such stochastic functions can become difficult for complex natural systems. Multiple-point Statistics do not require to define such function and will instead infer it in an implicit way from the training image provided by the user [Hu and Chugunova 2008]. The TI is a conceptual view of the geology or of the heterogeneity willing to be modelled. A TI does not have to honor specific data like borehole or outcrop measurements [Hu and Chugunova 2008], yet the TI has to describe the geometric patterns specific to the object aiming to be modelled. A TI can be either stationary or non-stationary. Stationary TIs are easier to use, they display a large repetition of patterns with a homogeneous spatial distribution. The non-stationary TIs generally include more information. These kinds of TIs need more pre-processing treatment and are coupled with auxiliary variable in order to be used in an efficient way by the algorithm (Figure 3). Even if the MPS appears to be powerful with the ability to rely on both conceptual TIs and hard data, it still has to overcome some difficulties when it

comes to reproduce natural objects. The geology of the subsoil is unlikely to be spatially homogeneous, but instead displays trends and heterogeneities. Due to the large variety of processes, it is usually unrealistic to represent the heterogeneity of the soil only with stationary information.

2.2.2 Non-stationarity Simulation

The main difference between classic statistics and geostatistics is the assumption of spatial dependency. This means that the location of data elements, plays an important role in the analysis and the modelling process. From a statistical point of view, a stationary process is defined by a constant mean, variance and autocorrelation structure over time or space. The covariance between a pair of points depends only on the distance between those two points and not on their location. This can be viewed from a more geological or geomorphological point of view as a zone free of any trends. Unfortunately, almost all the objects of interest in the hydrological or geological field (porosity, permeability, facies, and volumes) are created by processes implying heterogeneity and spatial dependency. These processes lead to the production of heterogeneous fields and non-stationary areas, layers or volumes.

An example of non-stationary field is represented by a fan delta (Figure 2). We can summarize the geomorphology of a delta as a main and large meander river (Figure 2-1) that splits into several smaller branches when approaching the sea (Figure 2-2 and 2-3). These branches can evolve to different shapes either straight shape (Figure 2-3) or anastomosed shape (Figure 2 -2). In this example, the rotation and scaling are non-stationary variables, since they display spatial dependencies over the fan. If we use this image as our TI, the non-stationarity that composes it would result in an unrealistic simulation. Without further information, the simulation could not honor the dispersion of the channels in the delta fan, neither the change of size.

When working with non-stationary TI, some rules have to be observed in order to produce realistic simulation. Since the repetition of patterns is not homogeneous in the TI, auxiliary variables are required (Figure 3). The auxiliary variable is used by the code to filter the sample location and thus, constrain the non-stationarity of the TI. With that information, patterns are not mixed together when simulated. Auxiliary variables can come from rotation maps, geophysical surveys or probability maps. It can be viewed as a continuous field overlapping the TI. During the simulation this field is then used to control the MPS algorithm. Consequently, only patterns associated with the auxiliary variables are simulated in the corresponding part of the domain.

An example of such simulation is presented on the figure 3. The TI represents a cross section that displays non-stationary information. The patterns are not homogeneously distributed and their repetition on the TI is low and heterogeneous. In order to obtain a good simulation output, it is needed to provide an auxiliary variable map that represents the non-stationary information present in the TI. In this case the auxiliary variable is a continuous variable map (Figure 3). We can see on the TI that the patterns follow a vertical trend. Creating auxiliary variable can sometimes be challenging and thus discouraged the use of non-stationary TI. Complex simulations can be created by using stationary TI coupled with secondary variables such as orientation or scaling maps. The modeller should always balanced the complexity of the process with the realistic expectation of the simulation.

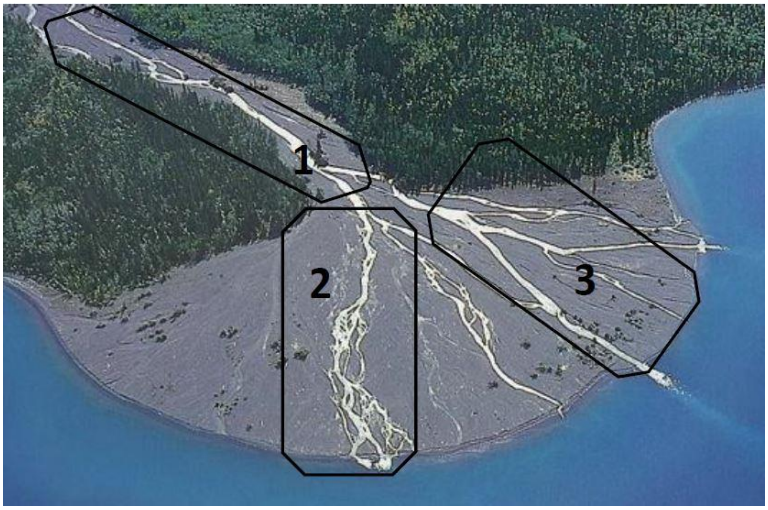


Figure 2 – Typical delta shaped river. Three main parts with different characteristics can be identified. This image shows non-stationary behaviour as the 3 parts display different shapes and orientations. 1) The main river with a straight or meander shape. 2) Small anastomosis shape channels. 3) Larger straight shape channels.

2.2.3 The Direct Sampling algorithm : DeeSse

The algorithm used in this Master research is the Direct Sampling Multiple-point Statistics algorithm (DeeSse) developed at the University of Neuchâtel by Prof. Philippe Renard and his team. During the last twenty years, many different MPS algorithms have been developed - Snesim [Strebelle 2002], Impala [Straubhaar et al. 2011], Direct Sampling [Mariethoz et al. 2010]. Each algorithm presents its own advantage and disadvantage regarding the data storage, data type and CPU requirements. DeeSse shows many advantages as it allows the user to work with categorical or continuous variables, to use multiple TIs during one simulation and to work with multiple-variables [Meerschman et al. 2013].

DeeSse is an implementation of the direct sampling method, where the TI is not scanned before the simulation, but randomly sampled as the simulation runs [Meerschman et al. 2013]. This method is statistically equivalent to other MPS methods, but offers the advantage of making database creation unnecessary [Mariethoz et al. 2010]. This method coupled with a distance calculation approach - the value of the mismatch between data in the simulation grid (SG) and in the TI - allows the algorithm to work both with categorical and continuous variables [Mariethoz et al. 2010]. With the use of DeeSse, it is also possible to give to the algorithm additional information in order to constrain the simulation. Rotation value or maps can be used just like proportion or affinity maps. These information help the algorithm to produce more realistic simulations based on the knowledge of the modeller. Moreover, DeeSse allows to apply a specific weight on these hard data in order to enforce patterns consistency in their neighborhood during the simulation [Meerschman et al. 2013]. I present here the basic workflow of DeeSse (Figure 4) and the different elements required for a simulation.

The first required object is a simulation grid (SG). We can create arbitrary volumes to simulate within the grid, these volumes are called regions. Regarding the use of hard data, they must be spatially located inside the SG. The second essential element is the TI that displays the pattern aimed to be reproduced. Once this two elements are present, simple simulation can be performed. Other data can be used by the algorithm, such as rotation map, trend map or affinity map. However, we here focus on the simplest case, one simulation grid and one TI, to explain how the algorithm processes. Classic MPS algorithms usually scan the whole training image before the simulation and save the probability of patterns occurrence either in a list or in a tree shape file. This method is very memory consuming because it requires to first scan every possible patterns in the TI and to store them. DeeSse is a direct sampling method, the main difference is that the algorithm avoids the usual pre-scanning and storing parts. Instead of looking at the probability of every pattern in a list, the direct sampling algorithm samples randomly the TI in order to find

a matching pattern. This method is less memory consuming and more flexible regarding the type of data and auxiliary variables that are usable. Moreover, the randomly scanning method is statistically equivalent to the probability list method.

At each step of the simulation, the algorithm chooses a point to simulate in the simulation grid and stores the values and positions of its n closer neighbors. The training image is then scanned randomly in order to find a similar pattern as the one selected in the SG. DeeSse calculates a distance value to compare the SG pattern with the TI pattern [Meerschman et al. 2013] (Figure 4). For categorical value, the distance corresponds to the average of the mismatch between each couple of points compared between the SG pattern and the TI pattern. The distance value takes 0 for a perfect match and 1 for a complete mismatch (Figure 4). For continuous variables, other mathematical definition of the distance is recommended to be used instead of simple average one [Mariethoz et al. 2010].

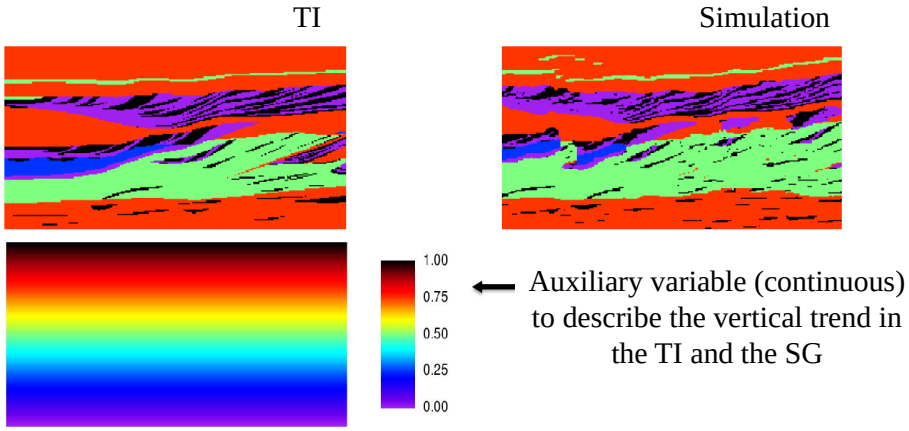


Figure 3 – Exemple of a MPS simulation using a non-stationnary training image. An auxiliary variable is created to constrain the simulation. The patterns are then well reproduced during the simulation. (Modified from [Comunian et al. 2011])

The three main parameters controlling the simulation are:

- n = the number of nodes taken for the pattern comparison,
- t = the acceptance threshold,
- f = the maximum scanned fraction of the TI.

They influence the simulation time and its quality. The n parameter defines the size and geometry of the pattern. At the beginning of the simulation, these n closer points are likely to be located far away from the simulation point. However, as the simulation progresses, the density of simulated point increases and the n closest points are likely to be located near the central point. This feature ensures DeeSse to reproduce structures of all size during the simulation, starting with large one and finishing with small and fine structures [Meerschman et al. 2013]. Since a perfect match is not always found, DeeSse proposes to use a parametric threshold value t. This threshold parameter is important regarding the simulation of continuous values where perfect matches are rarely found. If the distance calculated at the first random position in the TI does not respect the threshold parameter, another point is chosen randomly on the TI and the distance is re-calculated. Once the value of the distance has reached this threshold or that a perfect match is found between the data and the TI, DeeSse copies the value of the central point found in the TI into the simulation grid point. Another point is then chosen in the SG and the simulation goes on until all the points of the SG are filled (Figure 4). Finally the f parameter helps to reduce the simulation time while conserving realistic patterns reproduction. During simulation if a fraction f of the TI is scanned without finding a pattern satisfying the threshold condition t, the best scanned node (minimal distance) is retrieved. The same principles are used for continuous and multi-variables. The distance based method makes the algorithm very flexible, while the three parameters (n, t, f) allow the modeller to test rapidly its simulation, while controlling its quality.

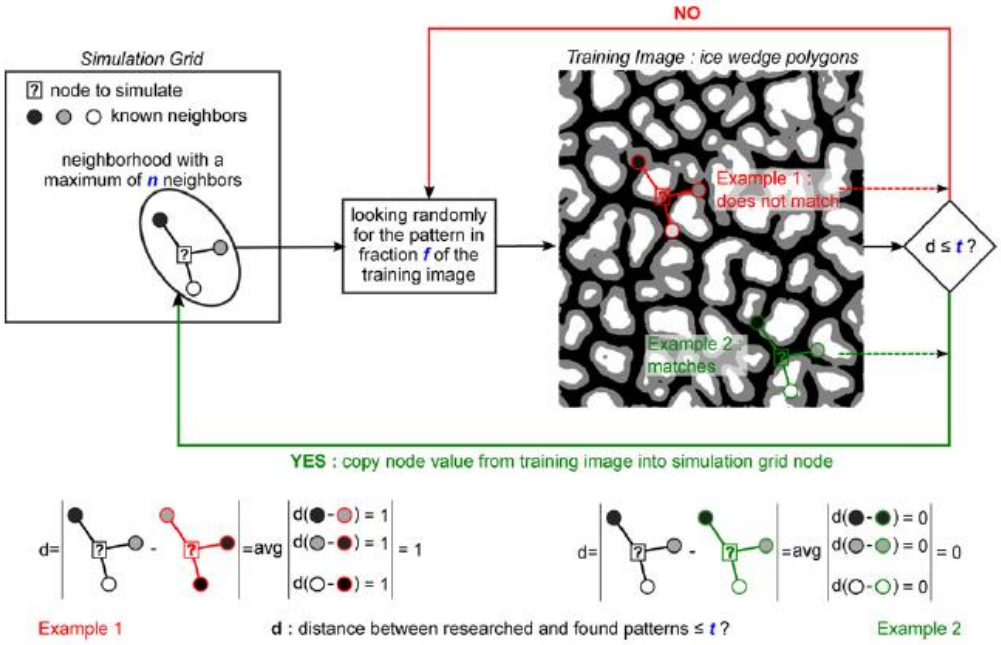


Figure 4 – DeeSse algorithm work path for the simulation of a three categorical values pattern expressed in the training image. The calculation of the distance value between the TI's data and the SG node is explained in the exemple 1 (mismatch of the pattern) and exemple 2 (perfect match of the pattern). [Meerschman et al. 2013]

3 Materials and Methods

I present in this section the different elements created for the MPS simulation and the basic concepts of the post-simulation tests realized in this project. The creation of the different auxiliary variables and the training image was an iterative task. During this project, I tested different trend maps, rotation maps and TIs in order to choose the most representative data for the simulation. Most of the auxiliary data were first tested in a 2D grid in order to understand their influence on the model. We developed new approaches for the creation of the different variables in order to create complex 3D auxiliary variables. The use of a non-stationary TI led to the creation of auxiliary variables maps (trend maps). Since we aimed to produce a 3D model with trends along the z-axis, we had to create a 3D auxiliary trend map. This implied the creation of a 3D trend, which took into account the horizontal and vertical trends present in the aquifer. Figure 5 summarizes the different elements created for the simulation, the different software used for their creation and the principal steps. All the different elements influenced the final model and were essential for the creation of the complex realistic 3D model (Figure 5). DeeSse was implemented in the geostatistical modelling software AR2GEMS [AR2GEMS 2010], where the different grids were created. We also used it for geostatistical calculation, data visualization and post-simulation data treatment. AR2GEMS allows to work with Python script, which provides a large flexibility for the modeller. In addition we use the ArcMap software [ArcMap 2016] for the geographic and spatial treatment of our different shape and raster files and the FEFLOW software [FEFLOW 2016] for the numerical flow calculation and the creation of the trend maps.

3.1 Simulation Grid

The first step was to create a simulation grid. I decided to first create a 2D simulation grid before creating the final 3D grid. This 2D grid was created in order to test the simulation parameters and the training images (TIs). Simulations in a 2D grid were easier and less time consuming and thus enable us to test different approaches. A shape file, corresponding to the limit of the project, was provided and used for the creation of the 2D grid. Then, I created two 3D grid corresponding to the PC. The first 3D grid correspond to the real volume of the PC aquifer, whereas the second 3D grid is transformed regarding the bottom elevation of the PC (flattened space). The second 3D grid was created for the simulation, since we have decided to work with a 2D TI and 2D simulations, where one simulation corresponds to a temporal sedimentation layer, we had to transform the grid in order to simulate inside layers of same depositional age. These two 3D grids were created from two raster files. These files were elevation maps of the top and bottom of the Pliocene Continental layer. These elevation maps had been created and provided by a partner of the Dem'eaux Roussillon project.

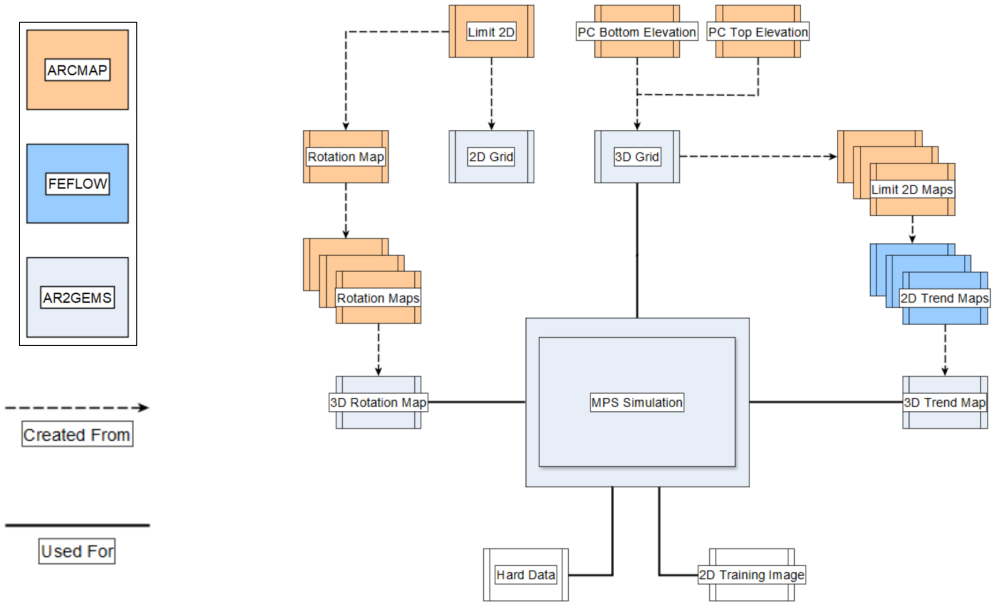


Figure 5 – Summary of the different steps and the variables used with DeeSse for the creation of the model.

3.1.1 2D Grid

The first step in the creation of the 2D grid was to define its dimensions and resolution. The dimensions of the grid had to be chosen according to: the size of the object aimed to be modelled, the resolution of the simulation and the computation limitation. After some tests and discussions, the 2D grid dimensions were set as: $407x504$ cells (205'128 cells in total), with a cell dimensions of $100x100m$. These parameters allowed to model the sedimentary objects regarding their extent, while maintaining a low simulation time. The grid was created by convert the shape file of the aquifer limits to a raster file with a binary property called "inside". The value 1 was assigned when a cell was located inside the simulation zone and 0 when located outside. We then imported this raster file in the modelling software in order to create our 2D cartesian grid and the corresponding simulation zone (Figure 6 - blue part). This 2D grid was a rectangular shape grid, based on the dimensions of the raster ($407x504$ and $100x100m$).

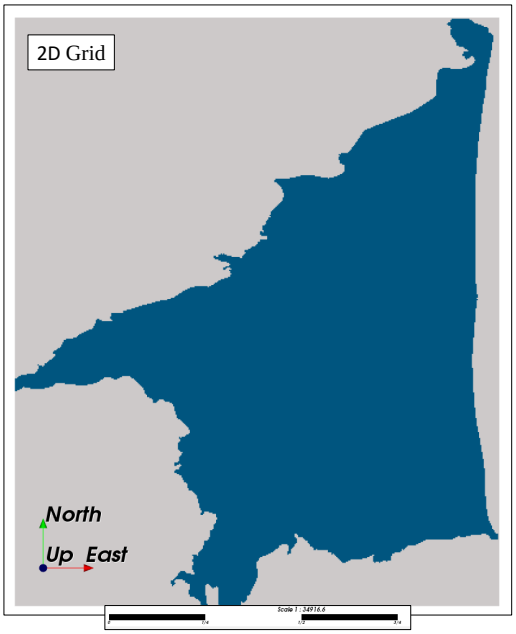


Figure 6 – The 2D simulation grid, the blue part corresponding to the simulation zone (region). The dimensions of the grid was $407x504$ and the cell size $100x100m$.

3.1.2 3D Grids

The process used to create a 3D grid was different than the 2D one. In order to create the 3D grid from the two 2D top and bottom raster maps, a Python script was created. First, the two 2D raster files were imported in the modelling software as two properties of the 2D grid. The 2D grid then possessed two properties for every cell, the maximal depth of the PC (the bottom property) and the minimal depth of the PC (the top property). The second step was to create an empty 3D grid in which the 3D PC volume was inferred from the two elevation properties of the 2D grid. The final 3D grid was defined with the following dimensions; $407x504x250$ cells (51'282'000 cells in total) with a cell dimensions of $100x100x2m$. The z-axis dimension was defined regarding the minimal and maximal value of the bottom and top properties. During the different tests, a first 3D grid with a $10m$ z-resolution was created to test the parameters. We then chose to increase the z-resolution, from $10m$ to $2m$ in order to use the maximum information available from log description. The last step was to apply a Python script. The script went through every cell in the 3D grid and compared the cell depth to the bottom and top depth properties of the same cell, in the 2D grid. If the 3D cell depth was below the value of the bottom property, the code assigned it a value 0 that can be interpreted as "outside the PC volume". If the cell depth was above the bottom value and below the top value, the code assigned it a value 1 that can be interpreted as "inside the PC volume". Finally, if the cell's depth was above both the bottom and top value, the code also assigned it a value of 0. A list was then created and the calculated binary value implemented at each cell evaluation. At the end, the code assigned the list to a new property of the 3D grid. A new region corresponding to the PC volume was then created inside the 3D grid based on this new property. This region defined the simulation zone of our 3D grid and corresponded to the volume of the aquifer (Figure 7 a). The Python script is available in the Appendix A.1.

Finally, a last 3D grid was created from the previous 3D grid. We called this new grid the transformed grid. It was created based on the topography of the bottom of the PC. This grid was transformed in order to work with 2D simulations that represent identical deposition age. This approach required to simulate inside layers of same geological periods, and thus required to transform the 3D grid regarding the topography of the PC bottom values (Figure 8). A Python script was created to transform the grid (Appendix A.2). This script first scanned all the positions of the 3D grid and computed from it a 2D index. This 2D index was equivalent to the number of cells by which a cell had to be shifted on the z-axis. When multiplied by the size of the z-cell dimension, this index was the depth that the code must subtracted on a cell, in order to create the transformed grid (Figure 7 b). This 2D index was also used to transpose the location of the hard data into the transformed grid.

The last function of this Python script was used to transform back the grid after the MPS simulations.

3.2 Hard data

In order to constrain the simulation and enhance the reality of the model, the use of hard data (or conditioning data) is recommended. In MPS simulations, hard data correspond to cells with an assigned value in the simulation grid. In our project, the hard data represent cells where the facies property was defined. This information was inferred from borehole data. The hard data used in this study were 52 well logs that had been described and interpreted. The 52 analyzed wells were mainly located on the Northern part of the Roussillon Plain (Figure 9). These wells were digitized, vectorized and interpreted in terms of facies deposits within the geological modelling software PETREL [Petrel 2017]. I interpreted the facies regarding their gamma-ray and resistivity response curves. The wells were grouped into cross-sections during the facies interpretation (Figure 9).

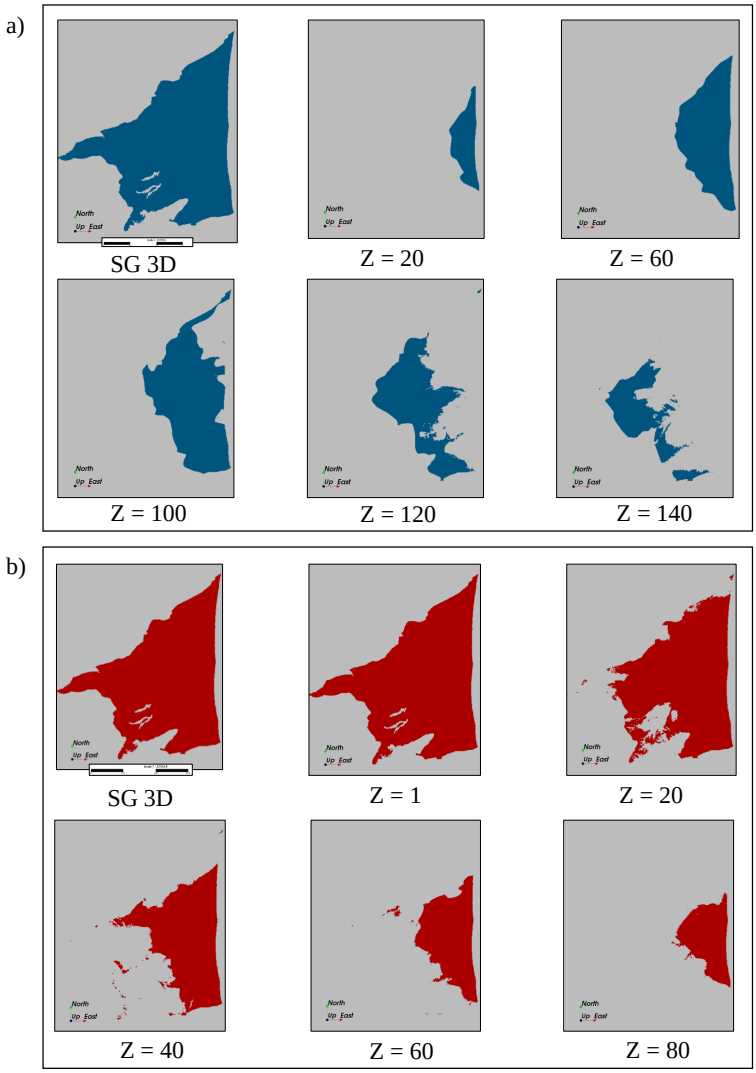


Figure 7 – 3D simulation grids and 2D layers at different z-positions (Z=1 is the bottom layer of the 3D grid). The first caption of (a) and (b) correspond to a top view of the 3D grid. The figure (a) corresponds to the 3D grid inferred from the top and bottom properties of the PC. The figure (b) corresponds to the 3D transformed grid as shown in figure 8.

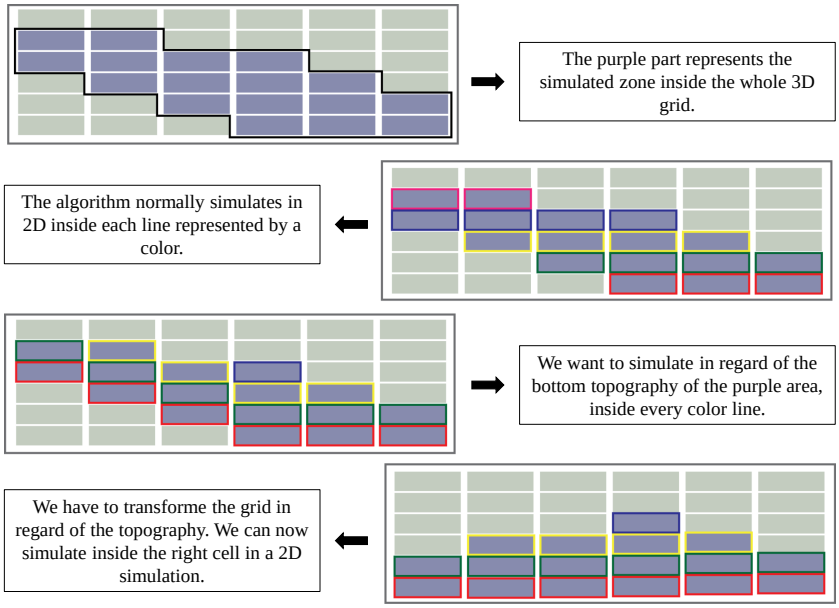


Figure 8 – Vertical cross-section of a 3D grid that is transformed regarding the bottom altitude of its simulated zone.

We have previously described that 6 facies were identified in the outcrops. Due to the small size of the levee deposit, this facies was left aside of the interpretation. The gamma ray and resistivity logs allowed to see the change in sedimentary deposits and grain distributions with depth. Sand sediments have small gamma ray response producing small peaks on the curve, whereas clay sediments produce high response peaks in the curve due to their high content in radioactive elements [Serra et al. 1975]. Fine sediments such as clay have low resistivity response due to their grain size and water content [Serra et al. 1975] whereas sand sediments tends to have a higher resistivity value. These two features allow to distinguish the different deposit types along the borehole. For example, a meander river deposit is associated with a half-bell shape curve in the gamma-ray response curve (10 - purple facies). The sediment are at first coarse sand deposits associated with low gamma ray response, which increases as the grain size decreases. Following these facies response behaviour, we interpreted the 52 well logs (Figure 10). We finally exported the data in text file. The resulting 52 text files were transformed using a Python script (Appendix A.3), in order to be used as hard data by DeeSse. The hard data input file must be discretized by points. Each point in the file must be described by its latitude and longitude, its depth and its property (facies) for every meter. The code extracted the facies information and

grouped them into one final text file. Moreover, it also collected, the longitude and latitude of each borehole and copied the information to every discretized points on the final file. Finally, the depth of every point was transformed to correspond to the transformed grid.

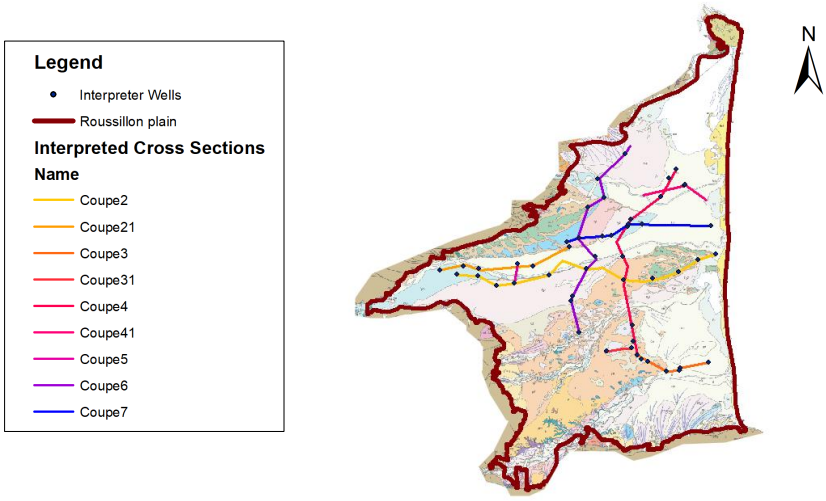


Figure 9 – Location of the 52 analyzed wells over the Roussilon Plain. The well logs have been interpreted along 9 cross sections.

3.3 Training Images

In order to create the TI, we first had to decide, which facies were the most relevant for this model. From outcrop observations and well logs analysis, six sedimentary facies were identified (the codes were used for their simulation) :

- Facies code 0 : Floodplain sediment
- Facies code 1 : Braided river sediment
- Facies code 2 : Meander river sediment
- Facies code 3 : Crevasse splay sediment
- Facies code 4 : Alluvial cone sediment
- Facies code 5 : Levee sediment

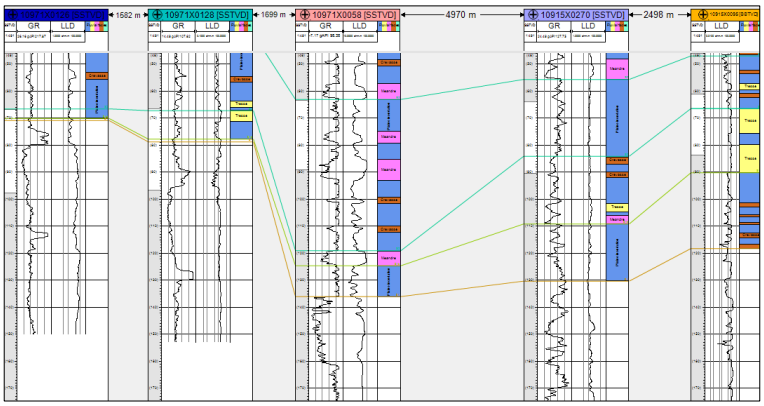


Figure 10 – Caption of an interpreted cross section composed of 5 well logs. The blue color corresponds to the floodplain deposit (code 0), the yellow color to the braided river deposit (code 1), the purple color to the meander deposit (code 2), the brown color to crevasse splay deposit (code 3) and the green color to the alluvial fan deposit (code 4).

Due to the large number of facies and the complex sedimentary processes that were associated, we chose to use a non-stationary training image. By using a non-stationary TI , we were able to express all of the deposit types in one TI (Figure 11 e). Different TIs had been constructed and tested (Figure 11 a-d). The TI (a) was the first TI tested, after the simulation we decided that the number of crevasse splay deposit had to be increased. The TI (b) was used to test large crevasse splays. The third TI (c) proposed a new connectivity pattern between the alluvial fan and the braided river and used only five facies (levee and crevasse were combined). Finally, we proposed to add crevasse splay deposit around the braided river, following the idea that the presence of the floodplain must be combined with overflow structure (Figure 11 d). These TIs had been tested in the 2D simulation grid. The final training image chosen for the simulation represented the best combination of the previously tested TIs. The final TI (e) was composed of 6 different facies. The alluvial fan covered the entire left part of the TI, the braided river start from every part of the fan, the crevasse splay and the levee are separated and the extent of the crevasse splay had been increased. This image had been laterally duplicated in order to control the lateral connectivity of the deposits. The final dimensions of the training image was $100x125$ with cell size of $100x100m$. These dimensions had been chosen in order to avoid affinity transformation during the simulation.

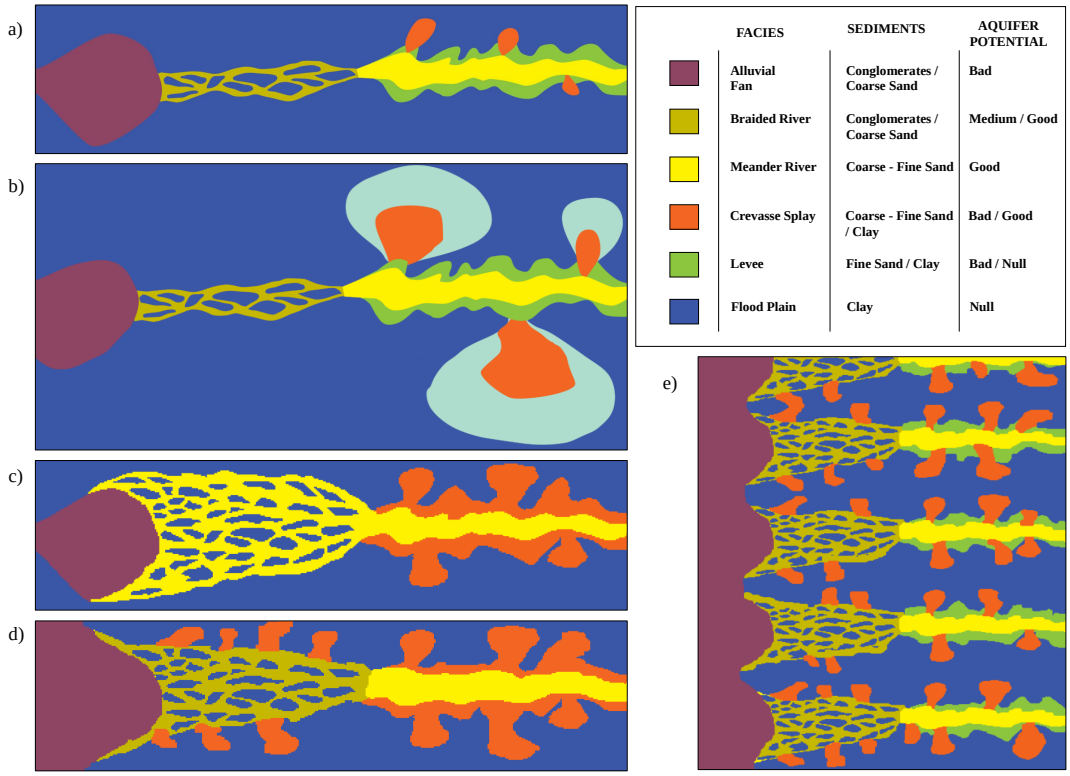


Figure 11 – Training images used for the simulation. The TI form (a) to (d) were used for the test of the facies descrition. The final TI used for the model was the TI (e) ($100x125$, $100x100m$). All of the TIs were made using a free drawing software.

3.4 Trend Maps

After the creation of the TI, the following step was to produce the auxiliary data. We had to inform the algorithm where the patterns of the TI had to be reproduced in the SG. Two auxiliary maps (trend maps) had to be produced, one for the TI and one for the simulation grid. These two trend maps had the same data range. These data served to guide the simulation, if a simulated node of the SG had an auxiliary data value of "0.4", the algorithm would search a corresponding node in the TI that satisfied both the auxiliary data and the pattern consistency. The trend map for the TI was produced with AR2GEMS, which allowed to create simple auxiliary properties that shared the same extent than the corresponding grid. The TI trend map was a simple gradient going along the x-axis (Figure

12). It started from the right with the 0 value and goes to 1 on the left side of the TI. Other trend maps for the TI were tested. By using Python script, different auxiliary maps were created for the TI, where the gradient distribution was modified. After some test, we decided to let a liberty degree during the simulation for the gradient value (higher threshold value) and thus we used the standard gradient distribution for the simulation (Figure 12 a).

Creating trend maps for the simulation grid was complex and required to develop a new approach, different from the one used for the TI. Since the simulated area was not the entire grid, the AR2GEMS gradient was not used. Moreover, a linear gradient would not display the horizontal heterogeneity of the plain trend. A new approach was developed to create complex auxiliary maps for non-rectangular grid. We used the groundwater flow simulation software FEFLOW [FEFLOW 2016] in order to create the gradient map. The idea was to use a flow simulation calculation to generate a realistic auxiliary map that respected the shape of the plain and displayed the evolution of the sedimentation. We associated the evolution of the gradient value to the evolution of the hydraulic head between a source zone and an outflow zone. Since the TI represented the complete evolution of the sedimentation process in the plain, the trend map could be represented by the evolution of the hydraulic head through the plain between the sources (river input) and the outflow zones (the sea).

The first step was to create a simulation grid in the flow simulation software that represented the Roussilon Plain. We exported the contour of our 2D model as a shape file from ARCGIS and imported it in FEFLOW. We then generated an advancing front shape mesh, composed of 1850 elements, inside the limit defined by the shape file. The shape of the mesh and the number of elements that composed it did not influence the flow transport calculation because we only simulate simple hydraulic head calculation. Once the mesh was created, we define the ouput zone of the grid. The output zone corresponded to the sea shore of the plain, where we fixed a boundary condition (BC) of $0m$ for the hydraulic head parameter. Then we had to define the input zones of the plain. We defined four input zones corresponding to the three main rivers of the actual plain and one zone corresponding to the south relief (Figure 13). We assigned a BC hydraulic head of $100m$ to these zones. In order to produce a continuous trend, the input zones were enlarged comparing to the real river entrance positions. The hydraulic head value chosen for the simulation did not have a physical meaning, it was chosen to produce a homogeneous trend that reproduced the morphology of the plain. After the boundary conditions were defined, we ran the flow simulation as a steady states saturated groundwater-flow. We finally used the output of the flow simulation to create our trend map. The hydraulic head fringes computed in the mesh were exported as a shape file to be post-treated in ArcMap. The hydraulic head fringes were represented by polygon objects in the shape file. We used ArcMap to modify the

property of the fringes. We transformed the hydraulic head value of the polygon fringes to a gradient field going from 0 to 1 (Figure 12). The last step was to convert the shape file to a raster file and to import it into AR2GEMS for the MPS simulation.

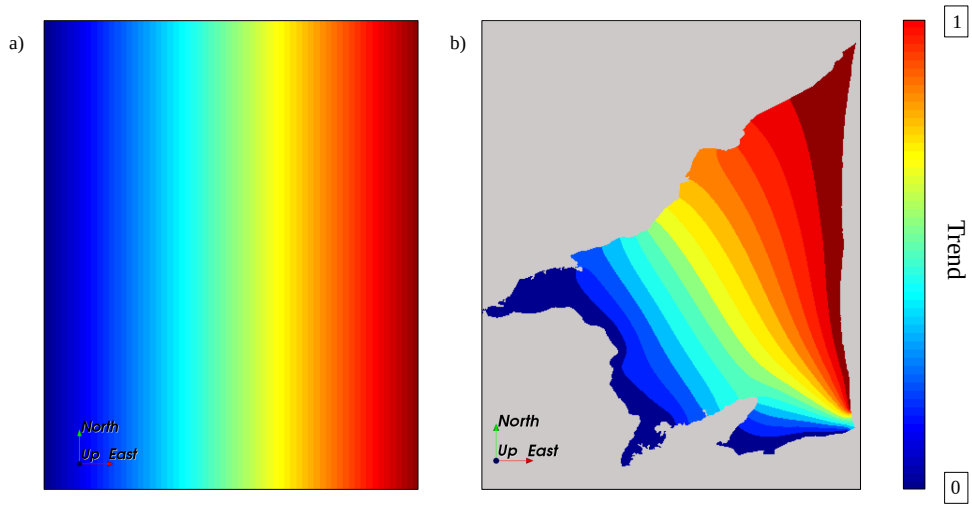


Figure 12 – 2D auxiliary variables for the TI (a) and for the 2D simulation grid (b).

The creation of the 3D auxiliary map used the same method that the 2D auxiliary map. We repeated this 2D approach for 11 representative layers before stacking these maps together to create a 3D auxiliary map. The first step was to extract the layers from the 3D grid and to apply the auxiliary map creation process to each one of them. After transformation, the 3D SG was composed of 110 layers, simulating an auxiliary map in each one of these layers would have been time consuming. Since the shape of the 2D layers did not change rapidly between layers (the layers 24 is almost the same as the layer 28 or 29), 11 representative layers were chosen to represent the different auxiliary maps. This technic allowed us to control the progradation of the system. During the sedimentation, the plain was prograding toward the sea. This means that the patterns of the TI had to be moved toward the sea as we moved up on in the layers. This is illustrated on figure 14, where the blue part moves toward the right side of the trend map as the altitude of the layers increased. Once simulated, the auxiliary map of a selected layer was copied until the second simulated layer. We choose to simulate 11 layers; 0, 10, 20, 30, ..., up to 110. Once simulated, the layer 0 will be assigned to the layers 0, 1, 2, 3, 4, ..., 9, then the layer 10 to the next 10 layers and so on until all the layers had an auxiliary map affected. This was programmed using a Python script (Appendix A.4). With this technic, a complex 3D auxiliary grid was

created, accounting for the river input zone, the horizontal trend and the vertical trend of the sedimentation process of the plain.

3.5 Rotation Maps

The last auxiliary map created for the simulation was the rotation map. By informing the algorithm of the channels orientation, a better and realistic simulation was aimed to be produced. DeeSse allowed to use a continuous rotation parameter for every cell, which is not possible with other MPS algorithms [de Carvalho et al. 2017]. This feature enabled to use realistic rotation map with smooth transition zones. The rotation map provided information about the channels' orientation on the plain. This map was created from direct field observations and from the actual rivers orientation observed on the plain. The map was generated in ArcMap, the first step was to create orientation lines that covered the plain. We drew lines along the principal zone of the plain and had a rotation field for every line. The rotation field were values corresponding to the rotation of the pattern desired in the simulation. A positive value corresponds to a clockwise rotation and a negative value to a counter-clockwise rotation of the pattern. Once created, the lines were transformed

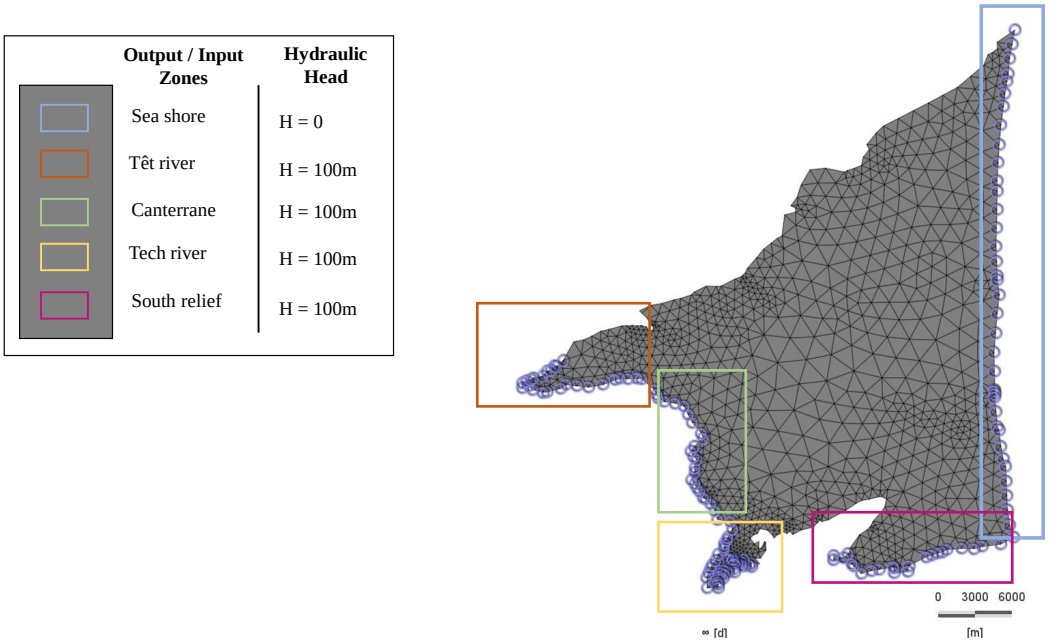


Figure 13 – The FEFLOW mesh and the different boundary condition zones.

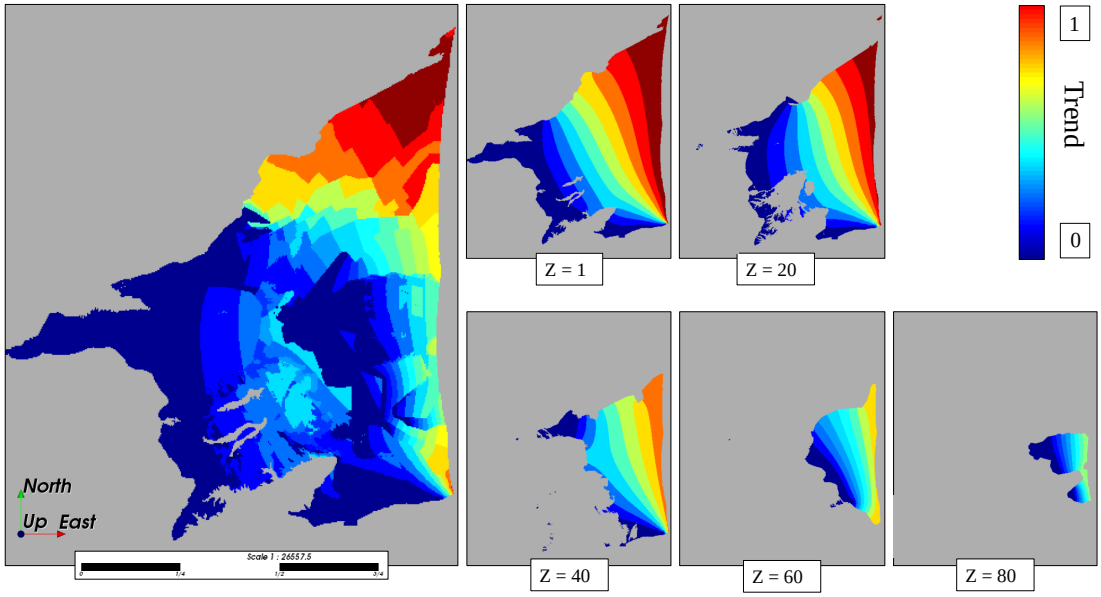


Figure 14 – The 3D auxiliary variable for the 3D transformed grid. The prograding sedimentation process was represented by the shifting of the blue fringes towards the sea. Both vertical and horizontal trends of the PC were integrated in the 3D trend map.

to points using the "points from lines" tool proposed in ArcMap. With this procedure, we obtained rapidly a large set of points with a rotation field that covered the entire plain. This set of points was then used to produce a kriging map of the rotation field. The kriging map was finally transformed to a raster file and exported (Figure 15).

DeeSse allows to work with two rotation maps that define the minimum and the maximum value of the rotation. This method allows the simulation to be more flexible on the rotation value. We modified the rotation map in order to create a second one with a rotation shift of 20°, which gives a liberty of 20°during the simulation. The last step was to create the 3D rotation map. We assumed that orientation of the channels were constant through time and thus identical over the different layers. The two 3D rotation grids were therefore generated by assigning the 2D rotation maps to every layers of the transformed 3D grid.

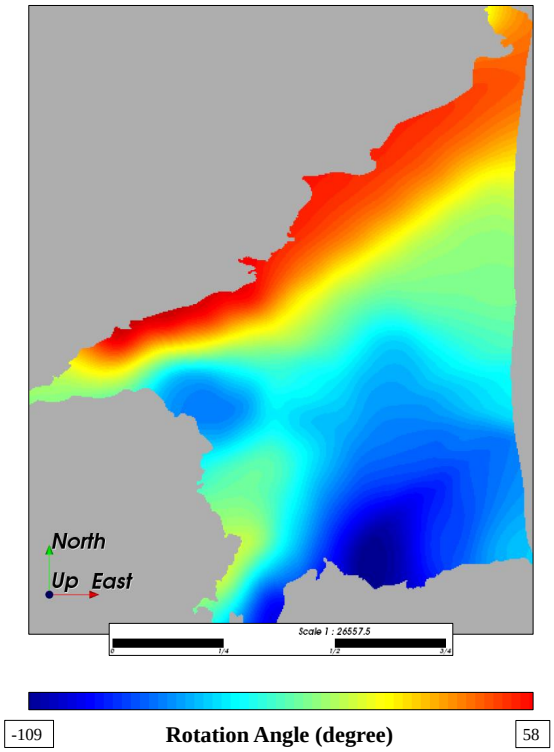


Figure 15 – 2D rotation map. The North area (red zone) corresponds to a clockwise rotation of the TI patterns. The central zone (greenish zone) corresponds to a zone where patterns are not rotated. The South zone and West zone (blue zone) defines a counter-clockwise rotation of the patterns.

3.6 Simulation Parameters

We present here the main parameters chosen for the simulation. The complete input parameter file is available in the Appendix section (Appendix A.5). Each parameter was tested and chosen in order to optimize the simulation time without impairing its quality. In our simulation we did not use the scaling ratio (affinity ratio) because the size of the patterns in the TI was already matching the size of the modelled objects. The rotation with tolerance option was used in our simulation. The tolerance was fixed by the two rotation maps previously created. Two variables were used during the simulation, the facies parameter (with the TI) and the trend parameter (with the trend map). We defined for these

two parameters, the search neighborhood parameter, the maximal number of neighboring nodes, the weight factor for the conditioning data and the distance threshold parameter as shown in Table 1. The maximal scanned fraction of the TI was set to 0.5 and no pyramid level was used for the simulation.

Once satisfied with the 3D simulation, the last step was to produce a large number of equally statistical simulations in order to compare them and studied their variations. This part was performed using the CPU cluster of the university, allowing to parallelize the computation between the CPUs and reducing drastically the simulation time. 100 simulations were performed, they all shared the same input data and parameters, the only change was the seed number of the stochastic calculation. A random list of numbers is calculated by the algorithm during simulation. These random numbers define the path in which points are simulated. It is important to be able to reproduce a simulation in order to control the different parameters. The use of a seed number and a random number list, allowed reproducibility in simulation.

Table 1 – Main parameters used for the MPS simulation with the DeeSse algorithm.

	Search Neighborhood parameter: -search radius in each direction -anisotropy ratio -rotation angle	**Maximal number of neighboring nodes in the search neighborhood**	**Weight factor for conditioning data**	**Distance threshold**
Facies parameter	20 20 0 1 1 1 0 0 0	24	5	0.05
Trend parameter	20 20 0 1 1 1 0 0 0	10	-	0.25

3.7 Probability Maps and Shannon Entropy

In order to understand and visualize simply the uncertainties that were considered in our model, we created a set of 100 simulations. The stochastic approach allows to generate sets of equiprobable outputs. Simulating a large number of realisation from our model enable us to calculate probability maps for every facies based on the set of simulations. These maps display the probability of a facies to be simulated at one location based on the simulations set. If a facies is well understand and largely constrain, it is likely that all the simulations will simulated this facies at the same location and thus the probability map will display a high probability of simulation. Howerver, if a facies is less understand and constrain, its probability map will display larger zone of occurrence through the simulations with smaller probability value. These maps helps the modeller to understand its model and its uncertainties.

We then used these proportion maps to calculate the Shannon Entropy of the set. The Shannon Entropy represents the amount of information that is included within a probabilistic distribution. The Shannon Entropy comes from the theory of information developed by Shannon in the middle of the 20th century [Shannon 1948]. It is a probabilistic theory that defines the amount of information contained in a data set. This theory was developed to estimate the maximum quantity of information that can be delivered by a certain source. It is also used to calculate the maximal encoding of a signal in order to avoid loosing any information. The quantity of information is equivalent as the amount of uncertainty contained in the simulation. Thus, when the data source has a low-probability value, the event carries more information than the source data which has a high-probability value. It can be viewed as a variance-like representation of the simulations, only for continuous variable. The formula of the Shannon Entropy is expressed as:

$$H(X) = \sum_{i=1}^{n} p_i * log_b(\frac{1}{p_i}) \tag{1}$$

where b is the logarithm base and p_i the probability of occurrence of the element i. The logarithm is usually in base two for the binary encoding use. We used the base six in our calculation which corresponds to the number of different facies in the simulation. The entropy is by definition, maximal when all the outcomes have equiprobable statistics and is defined as null when there is no variation in the source signal. I used a Python script to calculate the entropy of the set of 100 simulations (Appendix A.6). The script extracted

the probability of every outcome from the proportion maps, then calculated the entropy and created a new property to the 3D SG. The value of the calculated entropy was only a qualitative information and it allowed us to determine the zones of high entropy or large uncertainty.

3.8 Transmissivity Estimation

Finally, we tried to calculate the hydraulic conductivity of the simulated facies. 180 transmissivity values were measured over the Roussillon Plain. I used this set of data to calculate the hydraulic conductivity of the simulated facies. Two approaches were tested. The first one implied the use of a linear regression analysis using an inverse matrix calculation. The second approach used an optimization algorithm in order to minimize the error between the logarithm measured transmissivity and the logarithm calculated transmissivity, while adjusting the hydraulic conductivity values. I used a Python script to extract the facies of the simulation and to test the two approaches (Appendix A.8).

I first tested a least square estimation of the hydraulic conductivity. It required to get the conductivity value of the six simulated facies in respect to the measured transmissivity location. The transmissivity is calculated from the permeability by the folowing equation:

$$T_i = \sum_{j=1}^{6} e_{ij} K_j \,, \quad i = 1, ..., 180 \tag{2}$$

- T_i : transmissivity measured at point i, $i = 1, ..., 6$
- K_j : hydraulic conductivity of facies j (K_j unknown), $j = 1, ..., 6$
- e_{ij} : length portion of a facies j along the vertical axis at point measurement i

Once written in the matrix form, the matrix dimensions become:

$$\mathbf{T}_{180,1} = \mathbf{E}_{180,6} \cdot \mathbf{K}_{6,1} \tag{3}$$

After some matrix transformation, the final system that was calculated corresponded to:

$$\mathbf{K} = [\mathbf{E}^t \cdot \mathbf{E}]^{-1} \cdot \mathbf{E}^t \cdot \mathbf{T} \tag{4}$$

The second approach used a optimization calculation in order to determine the best hydraulic conductivity parameters that minimize the error between the logarithm measured transmissivity and the logarithm calculated transmissivity:

$$E = \sum_{i=1}^{n} (log(Ti_{measured})^2 - log(Ti_{estimated})^2) \tag{5}$$

The optimization algorithm tried to minimize this objective function, by adjusting the input parameters of the function - the hydraulic conductivities. Such method allowed to define bounds of values for the input parameters. Classic optimization algorithms are usually based on gradient slope analysis. The chosen optimization algorithm was the Sequential Least squares Programming (SLSQP) optimizer from the *scipy* Python library. The Least square method is a classic approach in analyse, used in order to approximate a solution of an overdetermined system. This optimization method was coded in a Python script (Appendix A.8). At each step of the optimization the algorithm aimed to reduce the objective function and so the error between calculated and measured transmissivity. This SLSQP method works with both linear and non-linear function which makes it very flexible.

4 Results and Discussions

After presenting the different variables created for the MPS simulation, we now present the result of the multiple-variables model performed with the DeeSse algorithm. A first 3D model has been simulated and is now presented. Using the same parameters, a set of 100 simulations was performed. We used this set to calculate probability maps of the six facies and we estimated the amount of uncertainty of the model by using the Shannon entropy. Two other sets of simulations were also created and analysed in order to understand the influence of the distance between meander bed in the TI, these two sets are presented in the Appendix A.9 - A.10. Vertical proportion curves (VPC) were created and are presented in this section. We used the VPCs to provide a visual validation of the model by comparing the VPC of the hard data with the VPC of the simulations. Finally, an inverse validation method has been tested. We inferred the hydraulic conductivity of the different facies from transmissivity measurements. We then calculated back the transmissivity and compared the difference between the measured and the calculated transmissivity.

4.1 3D Mutiple-Variables Simulation

Our 3D model was created by using all of the previous auxiliary maps, which reflected our knowledge of the sedimentation processes of the plain. The output of the MPS simulation is presented on figure 16. The simulation was performed inside the transformed grid. The simulation happened to reproduce well the training image patterns over the different layers of the grid. Moreover, the prograding process of the sedimentation plain was well simulated and was visible with the progression of the alluvial fan facies through the upper layers. With the addition of a liberty degree to the trend variable, the simulation increased the hard data and patterns consistency, while honoring the rotation maps. The multiple-variables approach combined with a complex rotation map was a success, we produced a large and complex realistic model of the Continental Pliocene aquifer.

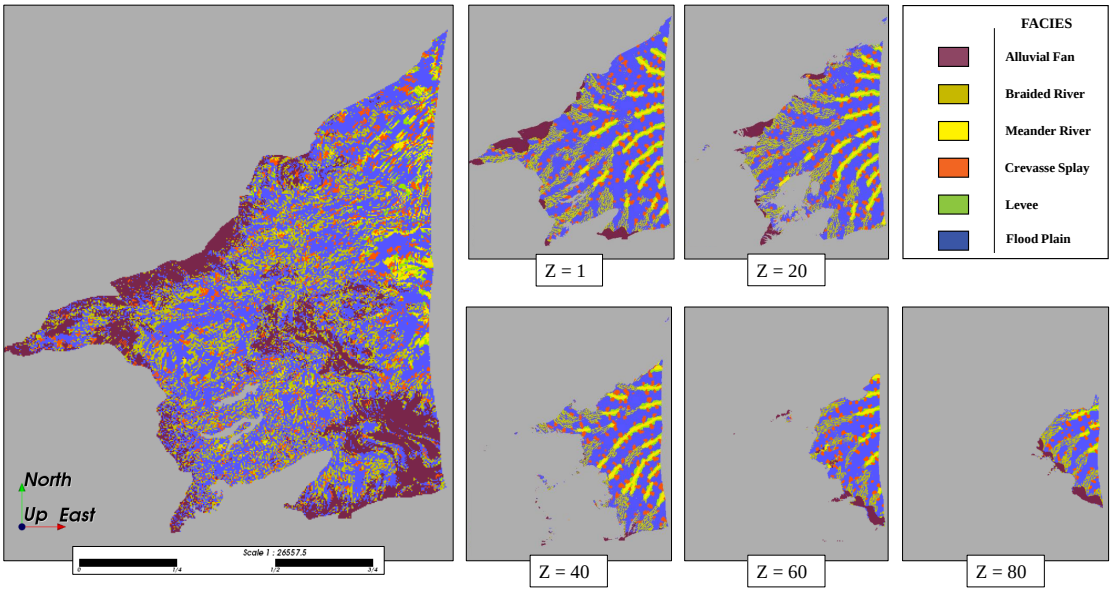


Figure 16 – The 3D multiple-variables MPS simulation. The first left figure represents a top view of the 3D model. The other figures represent different simulated layers of the transformed 3D gird. Both vertical and horizontal trends of the plain were successfully simulated.

We then transformed back the 3D simulation to the initial 3D grid. The cell vertical dimension of this grid was multiplied by a factor 100 in order to study the connectivity of the layers and the overall aspect of the simulation (Figure 17). The connectivity between the different elements of the layers was not controlled during the simulation other than by the use of hard data. The simulation presented different vertical connectivity zones, the South part of the simulation seemed to be more vertically connected, compared to the North part of the plain. This could be explained by the presence of a higher number of hard data in the North plain. When compared to the geological cross section of the plain, the connectivity of the South part seemed over-estimated.

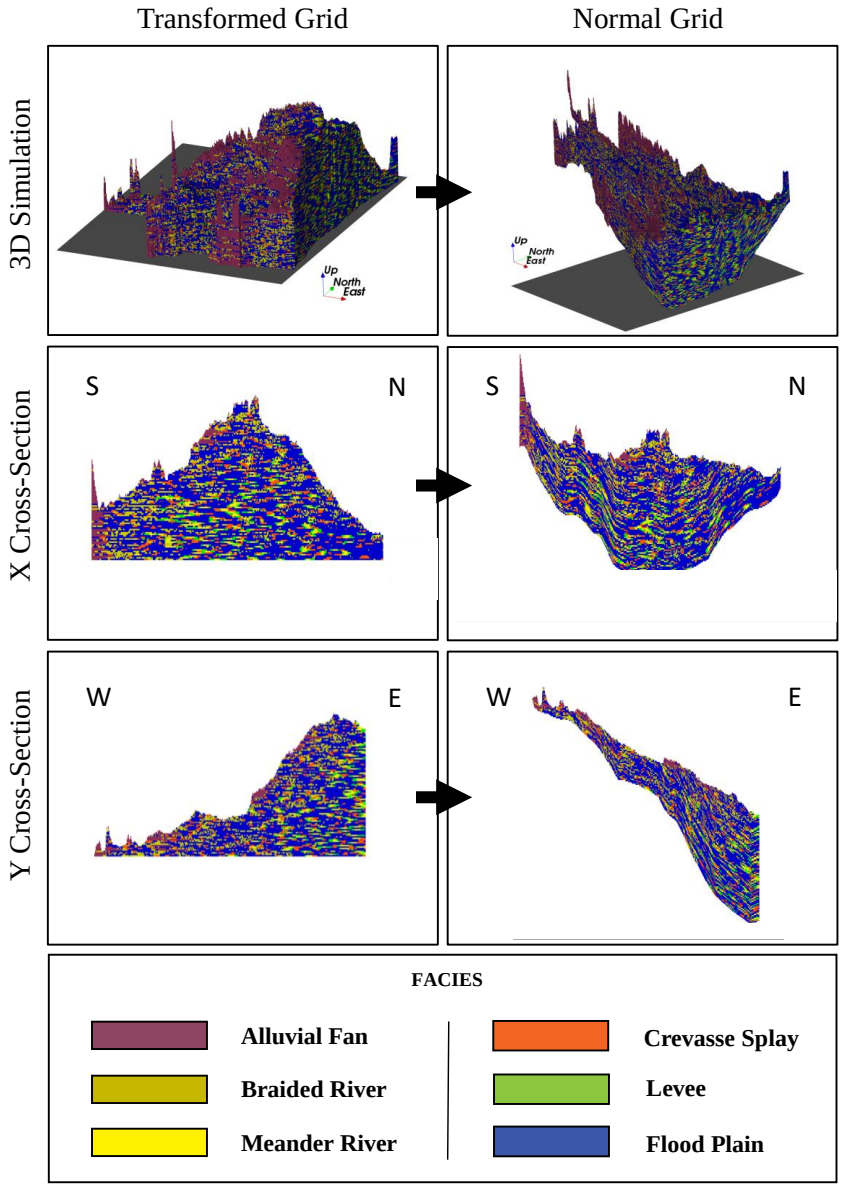


Figure 17 – 3D MPS simulation, the 3D grids are vertically deformed by a factor 100. The first column corresponds to the transformed grid, while the second corresponds to the normal grid.

The model succeeded to honor the patterns and trends reproduction. The new multiple-variables approach used in this project yield to the creation of a complex 3D model of the Roussillon Plain. The trend and rotation maps have successfully transferred the geological knowledge of the Plain to the algorithm. However, a higher number of hard data more homogeneously spatially distributed would have more constrained the vertical connectivity of the layers. Another approach could have been to define vertical limits where the vertical connectivity could had been constrained.

4.2 Probability maps and Shannon Entropy

We then proposed to study the set of 100 simulations in order to determine the zones of high variability. After the set of simulations was performed, we were able to produce post-simulation proportion maps for every facies (Figure 18). These proportions were calculated from the set of 100 simulations and allowed to determine for each facies the spatial variability of pattern reproduction based on our model. Some facies appeared to be more spatially constrained on the plain than others. The alluvial fan deposit presented very small spatial heterogeneity. It also appeared that the meander deposit presented a small spatial heterogeneity between the simulations. Since the meander facies was not well represented on the well logs analysis and that we allowed a 20°of liberty for the rotation parameter, we were expecting more variation in the spatial distribution of this facies. It appeared that the distance between each meander bed was too constrained by the TI. This was proved by performing two other simulations sets. One set with a TI that did not show lateral repetition and another one with the normal TI but without hard data. Even if the probability maps produced by the set of simulation without lateral repetition seemed to produce more variations in the meander spatial distribution, the facies proportion and the visual aspect of the simulations were not conclusive (Appendix A.9 - A.10). More work needs to be done to better understand the distance betweeen the meander channels and their lateral variations.

The calculation of the Shannon Entropy enables to group the information from the six proportion maps inside a single map (Figure 19). It appeared about the meander facies deposit and the central part of the plain from this analysis that more information need to be gathered. Another possibility is that the shape and dimensions of the meander bed in the TI are not representative of the reality of the deposit and need to be redefined.

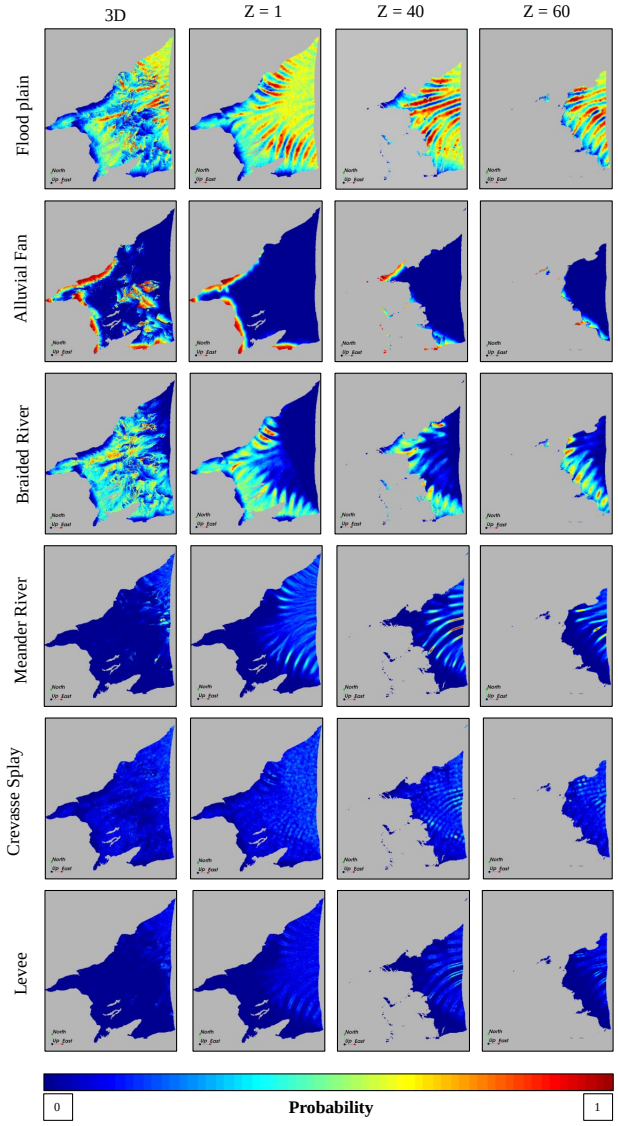


Figure 18 – Probability maps of every facies, calculated from a set of 100 statistically equivalent simulations. It appeared that the meander bed displayed no variability in their spatial location and was possibly too constrained in the model.

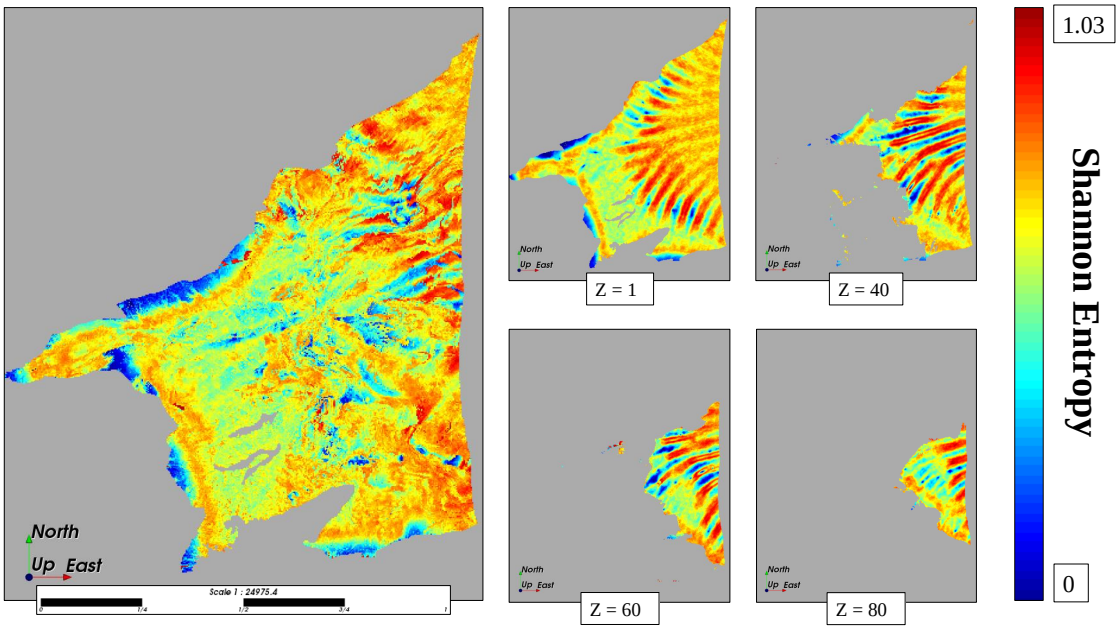


Figure 19 – Shannon entropy of the model, based on the proportion maps calculated for every facies.

4.3 Vertical Proportion Curves

We calculated vertical proportion curves from the well logs data and the simulations. This approach was used to assess the quality of the simulation [Ravenne C 2002]. We tested the reproduction of the vertical proportion trend by comparing the VPC of the well logs and the VPC of the simulation. Due to the small quantity of data and their spatial distribution, the wells VPC could not be quantitatively representative of the whole aquifer.

The VPC represents the facies proportion corresponding to a horizon in the aquifer. We transformed the depth of the well data into a relative depth calculated in regard of the bottom elevation of the PC. The first proportion corresponded to the bottom layer of the aquifer. We used a Python script (Annexe 6) to extract the proportion information from the simulation and to calculate the horizontal proportion of every layer. Two VPCs were produced from the simulation, the first one with a proportion normalized for five facies, the second one with the six facies proportion (Figure 20). The VPC of the hard data was created following the same transformation (Figure 20). The created VPCs allowed to read the evolution of the facies' proportion from the bottom to the top of the PC.

The VPC of the hard data seemed to underestimate the proportion of meander river compared to the field observations. Moreover, we should find alluvial fan deposits through the entire height of the aquifer. It appeared that the simulation reproduced more alluvial plain facies and more meander facies than the hard data. This was a good output of the simulation, since it was known that these two facies were under-estimated in the hard data. The floodplain proportion was reproduced with success by the simulation just as the crevasse splay facies. Finally we observed on both simulation and hard data, a shift in the sedimentation of the plain around 180m. This shift could be linked to the erosion of the plain by Quaternary river or by a miss-interpretation of the top limit of the PC.

The creation of VPCs was a useful tool during the development of our model, the test of the parameters and the final quality assessment of the simulation (Appendix A.7). It is not always easy to visualize a large 3D model and to understand its proportion distribution. The comparison between hard data's VPC and simulation's VPC was also interesting but should have included more well logs data. Despite the 52 wells used in this approach, the size of the plain required a higher number of interpreted wells. Moreover the distribution of the wells should be more homogeneous.

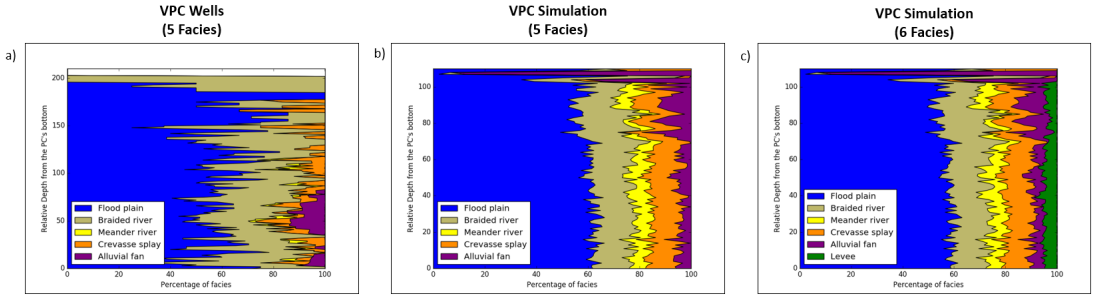


Figure 20 – Vertical proportion curves calculated from the hard data (a) and the simulation with five facies (b) and six facies (c). The horizontal axis corresponds to the facies percentage. The zero altitude corresponds to the bottom of the Continental Pliocene. For the simulations VPC, we specify that the vertical axis refers to the number of layers in the 3D model. Thus, the vertical depth has to be multiplied by the cell dimension (2m) in order to be compared to the wells VPC.

4.4 Transmissivity Estimation

In this last part, we propose to discuss the results of the transmissivity estimation approaches that were used to determine the hydraulic conductivity values of the sedimentary facies. The first approach was to solve an inverse problem, using the transmissivity data to calculate the hydraulic connectivity of the simulated facies. After the simulation, we extracted the different facies corresponding to cells location where a transmissivity value was known. Using a Python script, we then resolved the matrix inverse calculation. When resolving the equation system, the calculated values of hydraulic conductivity for the different facies did not have any physical sense (Table 2). Since the solution that satisfied this matrix system was not constrained by any bounds, the resulting hydraulic conductivity values included negative results. I calculated back the transmissivity from the hydraulic conductivity solutions in order to compare the measured and calculated transmissivity (Figure 21, blue cross). The plot showed that the calculated parameter did not succeed to approach the measured transmissivity.

The second approach was performed to overcome the negative value issue of the hydraulic conductivity and to improve the fit between calculated and measured data. This approach calculates the mean square error (MSE) between the logarithm of the measured transmissivity and the logarithm of the calculated transmissivity by using an optimization algorithm. The bounds were defined in order to keep the values of hydraulic conductivity positives. The optimization process also required as entry parameter a list of initial values for the variables. After different tests on the initial variables, an optimal list was found : K0 = 0.005, K1 = 0.2, K2 = 0.25, K3 = 0.1, K4 = 0.05 and K5 = 0.01. During those tests, the bounds conditions were also tested. It appears that the best optimization calculation were found when the bounds were the least restrictive. The returned hydraulic conductivities were all positive. However, the hydraulic conductivity of the facies were still very different than the one expected (Table 2). For example, the value of hydraulic conductivity of the floodplain facies was higher than the one corresponding to the braided river or the meander river. The floodplain sediments are mostly composed of silt and clay which should have a lower conductivity value than the braided sediments that are composed of well sorted sands. When plotted against the measured transmissivity, the calculated transmissivity from the optimization approach did not succeed in reproducing the measured values (Figure 21). The second approach has also been tested with the set of 100 simulations, however the results were not conclusive either.

It appeared that the assumption we made that the sedimentary facies were represented by unique conductivity value was erroneous. It was impossible with this assumption to find a unique conductivity value for each facies that satisfied the calculation of transmissivity.

Even with the optimization of the MSE function, there was always a residual error that could not be corrected. The hydraulic conductivities of the facies must be described by a range of values and are likely to be heterogenous according to their spatial location. The use of other methods should be prescribed to determine the range of the hydraulic conductivity of the sedimentary facies.

Table 2 – Results of the K estimation (m/s) from both methods, the matrix inversion and the MSE optimization.

	Floodplain	**Braided river**	**Meander river**	**Crevasse splay**	**Alluvial fan**	**Levee**
K1 (10^{-5})	3.07	2.48	2.97	-3.94	-4.71	1.26
K2 (10^{-5})	3.35	0.73	0.28	0.57	5.79	0.23

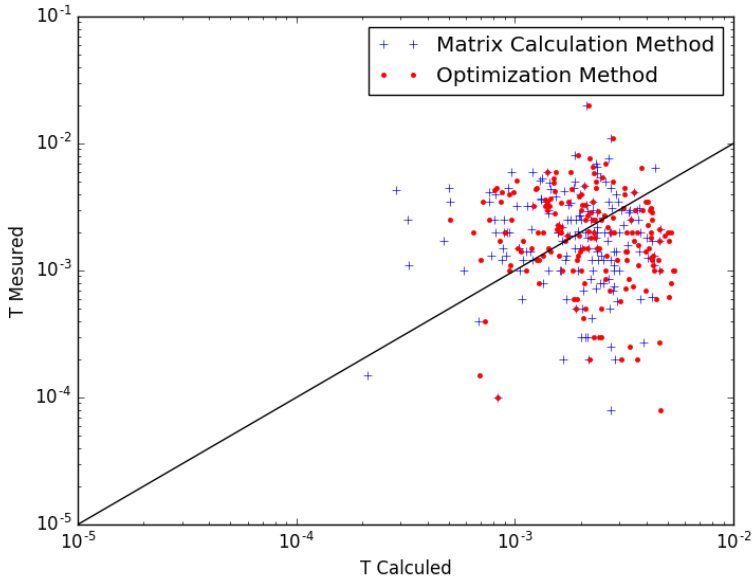


Figure 21 – Validation plot between the calculated and the measured transmissivity (m^2/s). The blue crosses represent the inverse matrix calculation and the red points, the optimization calculation. Both approaches did not succeed to estimate the K parameter in order to approximate the transmissivity value.

5 Conclusion

The aim of this project was to model the 3D geological heterogeneity of the Continental Pliocene aquifer of the Roussillon Plain. We could show that the Multiple-point Statistics approach and the DeeSse software allowed us to obtain plausible geological heterogeneity. This is the first application of this type of technique at that scale and for such type of aquifers.

The strength of the MPS technique and the DeeSse algorithm is that they offer multiple ways to model a geological system. The modeller can approach each problem with a different angle, which makes MPS, and especially the DeeSse very powerful and flexible.

Our work illustrates how these techniques can be applied using non-stationary multivariate simulations with the DeeSse algorithm. We used different approaches to manage the non-stationarity of the training image and constrain the model with 3D auxiliary variables. The final model shows plausible sedimentological patterns and trends reproduction while honoring the borehole data.

The robustness of the model has then been tested with a set of 100 stochastic simulations. From probability analysis and vertical proportion curves descriptions, it appeared that the model was missing sufficient borehole data to be fully constrained. The actual distribution of the hard data over the plain does not provide enough information to constrain all the facies. In particular, the meander river deposits appeared to be not fully understood, either in the training image description or in the borehole logs analysis.

Finally, we showed that we could not estimate the hydraulic conductivity of the different facies as constant values for each facies. The linear regression failed to provide fixed conductivity values that would allow to reproduce simultaneously all the observations. The hydraulic conductivity of the different facies is more likely to vary over the plain within the different facies. Other approaches should be used in order to determine their range of values. Permeability measurement from core logs along the plain could be used to calculate the hydraulic conductivity of the different facies and to study the spatial variation of this parameter. These permeability measurements, from core analysis, could be calibrated in a study site using a straddle packer to isolate the different facies horizons and then performing slug or injection tests.

The next steps to improve the final models would be to increase and harmonize, in terms of facies, the well log database of the Roussillon plain. The vertical connectivity, the vertical facies proportion trend and the vertical dimension of the objects are only constrained by the hard data. With an increase of hard data density, the quality of the model is likely to be enhanced.

Geophysical data could also be used to constrain the model and increase our knowledge regarding the meander bed. The general shape, the lateral connectivity and the distance between the beds could be understood and integrated to the training image to better constrain this facies.

Validation tests should also be developed for the simulation. Cross validation and vertical facies transition probability could be calculated to test the model. Moreover, the vertical size of the different objects could also be statistically calculated and compared to the log data.

One limitation to all of these approaches is the quality of the facies interpretation of the borehole logs and the spatial distribution of the data. Depending on the quality of the gamma-ray and resistivity logs and on the assumptions made for the interpretation, the quality of the validation process could be limited. For example, the decision to interpret the logs along cross-sections or individually is likely to result in different facies interpretation. The other limitation is about the size of the object modelled that cannot be controlled by DeeSse. It is likely that the size of the object will be realistically simulated near dense set of hard data but we don't know how far this information can be transferred laterally where the hard data network is less dense.

A Appendix

A.1 SG 3D Creation

```
# -*- coding: utf-8 -*-
import sgems

###################
###Juin 2018 Valentin
###################

#On récupère les dimensions du grid 3D
grid_geom = sgems.get_grid_info("SG3D")
nx        = int(grid_geom['num_cells'][0])
ny        = int(grid_geom['num_cells'][1])
nz        = int(grid_geom['num_cells'][2])

#Récupère les informations des altitudes mur et toit
z_bottom_name = "alt_mur_PC"
z_top_name    = "alt_toit_PC"
z_bottom      = sgems.get_property("SG2D",z_bottom_name)
z_top         = sgems.get_property("SG2D",z_top_name)

#On définie le nom des zones à créée; deux layers => 3 régions, l'aquifer 3D correspondera à la région 2
region1_name = "region1"
region2_name = "region2"
region3_name = "region3"

#Taille d'une couche du grid
nxy = nx*ny

#On stocke les propriétés d'altitude de toutes les cellules du grid 3D
z = sgems.get_property("SG3D","_Z_")

#On crée les couches par rapport à leur valeur z et à la valeur toit et mur de la même position
layer = [sgems.nan() for i in range(len(z))]

for k in range(len(z)) :

  if z[k] <= z_top[k%nxy] and z[k] >= z_bottom[k%nxy]:
    layer[k] = 1

#On assigne la nouvelle propriété créée contenant l'aquifère au grid 3D
layer3D_name = "layer_OK"
gridName     = "SG3D"

#On assigne les valeurs de la list layer dans la propriété layer_ok dans la couche SG3D
sgems.set_categorical_property_int("SG3D",layer3D_name,layer) #0

#On crée des régions pour chaque couches
for i,region_name in enumerate([region1_name, region2_name, region3_name]) :
  sgems.execute("SetRegionFromCategoricalPropertyIf {}::{}::{}::L{}".format(gridName,region_name,
  layer3D_name, i+1))

print "THE END"
```

A.2 SG 3D Transformed Creation

```
# -*- coding: utf-8 -*-

####################################
#Valentin Mai 2018
####################################

def topoToGrid(gridName,propertyName,valeurPropertyIn):

    import sgems

    ###
    #Récupère les info du grid 3D
    ###

    grid_geom = sgems.get_grid_info(gridName)
    nx        = int(grid_geom['num_cells'][0])
    ny        = int(grid_geom['num_cells'][1])
    nz        = int(grid_geom['num_cells'][2])
    grid      = sgems.get_property(gridName,propertyName)
    sizeLayer = nx * ny

    ###
    #Récupère toutes les couches en format une couche par liste dans une seule et même liste
    ###

    layer = []
    for i in range(nz):

        compteur1 = 0 + i * sizeLayer
        compteur2 = (i + 1) * sizeLayer
        layer.append(grid[compteur1:compteur2])

    ###
    #On crée une carte d'altitude ainsi qu'une liste de liste, contenant toute les colonnes
    #de la couche applatie
    ###

colonne = []
alt     = [-1 for u in range(nx * ny)]

    for i in range(nx * ny):
        z = []

        for j in range(nz):
            layerZ = layer[j]

            if layerZ[i] == valeurPropertyIn:
                z.append(1)

                if alt[i] == -1:
                    alt[i] = j

        if len(z) != nz:
            while (len(z) != nz):
```

```
                z.append(0)

        colonne.append(z)

    ###
    #On crée un nouveau grid à partir des colonnes précédentes,
    #transformé par rapport à la topographie
    ###

    gridTransfo = []

    for i in range(nz):

        for j in range(nx * ny):
            z = colonne[j]
            gridTransfo.append(z[i])

#On crée la nouvelle propriété grid transformé,
#c'est dans ce grid que les simulations vont avoir lieux
    sgems.set_property('SG3D', 'GridTransfo', gridTransfo)

    print ("Fonction topoToGrid terminée")
    return(alt,gridTransfo);

####################################
#Fonction qui re-transforme un grid avec la bonne altitude
#Valentin Mai 2018
####################################

def transfoBackToTopo(gridName,alt,gridTransfo):

    import sgems

    ###
    #Récupère les info du grid 3D
    ###

    grid_geom = sgems.get_grid_info(gridName)
    nx        = int(grid_geom['num_cells'][0])
    ny        = int(grid_geom['num_cells'][1])
    nz        = int(grid_geom['num_cells'][2])

    ###
    #On transforme le grid 3D en liste de couches
    ###

    layersTransfo = []
    positionGrid  = 0

    for i in range(nz):
        layerT = []

        for j in range(nx * ny):
            layerT.append(gridTransfo[positionGrid])
            positionGrid += 1

        layersTransfo.append(layerT)
```

```
    ###
    #On créer un grid 3D vide ainsi qu'une liste de couche de ce grid vide
    ###

    gridBack = [sgems.nan() for u in range(nx * ny * nz)]

    layersBack = []

    for i in range(nz):
        listeB = [sgems.nan() for u in range(nx * ny)]
        layersBack.append(listeB)

    ###
    #On retransforme les couches une à une
    ###

    for i in range(nz):
        layerZTransfo = layersTransfo[i]

        for j in range(nx * ny):
            b = alt[j] + i

            if str(layerZTransfo[j]) != 1:
layersBack[b][j] = layerZTransfo[j]

    ###
    #On crée le grid 3D retransformé avec les propriétés simulées
    ###

    gridBack = []

    for i in range(nz):

        for j in range(nx * ny):
            gridBack.append(layersBack[i][j])

    sgems.set_categorical_property_int('SG3D', 'BackToBack',gridBack )

    print ("Fonction transfoToGrid terminée")
    return;

###
#Appelle des fonctions et exécution du script
###

# -*- coding: utf-8 -*-
import sgems

gridName="SG3D"
property="layer_OK"
valeur=2

alt,gridOut=topoToGrid(gridName,property,valeur)
simuPropertyName="SimuWellTrendOrientationWeight1_real00000"
simuProperty=sgems.get_property(gridName,simuPropertyName)
transfoBackToTopo(gridName,alt,simuProperty)

print "THE END"
```

A.3 Well Data Extraction

```
# -*- coding: utf-8 -*-

import txtToPointSet as toP
import pandas as pd


########################
###Valentin JUIN 2018
########################

exportAllWell = []
nombreFile    = 52

for i in range (nombreFile):

    fileName   = toP.getFileName(i)
    coordonnee = toP.getCoordinate(str(fileName))
    facies     = toP.getFacies(str(fileName))

    for j in range(len(facies)):
        exportAllWell.append(str(coordonnee)+','+str(facies[j]))

############################
#Export le fichier csv final
############################

df = pd.DataFrame(exportAllWell,columns=['ID,X,Y,Z,PC,Profondeur,Facies'])
df.to_csv('exportPointSet.csv',index=False)

############################
#Supprime les "" et transforme le fichier
#Chaque ligne correspond désormais à une profondeur de passe de puits
############################

with open("exportPointSet.csv","r") as myFile :
    text = myFile.read()
    text = text.replace('"','')

myFile.close() #Ferme le fichier d'entré


with open("exportPointSet.csv", 'w') as my_file:
    my_file.write(text)

my_file.close()#Ferme le fichier d'entré


############################
#On transforme les données pour avoir les profondeurs transformées pour les simulations MPS
#La transformation s'éffectue par rapport à l'altitude réel du MNT et par rapport
#au numéro de couche transformée
############################

dataFinal   = pd.read_csv("exportPointSet.csv")
altitude    = dataFinal["Z"]
```

```
profPasse   = dataFinal["Profondeur"]
positionT   = dataFinal["PositionTransfo"]
profReel    = altitude-profPasse
profTransfo = profReel-(2*positionT)

dataFinal.insert(5,"profondeuReel",profReel)
dataFinal.insert(6,"profondeurTransfo",profTransfo)

dataFinal.to_csv("C:\Users\wdall\Desktop\DonneeForage\Geoter_Diagraphiesl\exportPointSet.csv",index=False)

print "THE END"
```

A.4 3D Trend Creation

```
# -*- coding: utf-8 -*-
import sgems

###
#VALENTIN MAI 2018
###

def getLayerTrend(indice):
import sgems

gridTrendName = "layer"+str(indice)+"trend_1"
propertyName  = "gradient"
#sgems.execute("ConvertToCategoricalProperty  {}::{}".format(gridTrendName,propertyName))
layer=sgems.get_property(gridTrendName,propertyName)
#sgems.execute("DeleteObjectProperties  {}::{}".format(gridTrendName,propertyName+" - categorical"))

print ("getLayerTrend Done")
return layer;


def getListe3D(gridName, propertyName):
import sgems

grid_geom = sgems.get_grid_info(gridName)
nx = int(grid_geom['num_cells'][0]) #nombre de cellule en x
ny = int(grid_geom['num_cells'][1]) #nombre de cellule en y
nz = int(grid_geom['num_cells'][2]) #nombre de cellule en z
SGProperty=sgems.get_property(gridName,propertyName)
sizeLayer=nx*ny
listLayer=[]

for i in range (nz): #On parcourt nz tour de boucle
compteur1=0+i*sizeLayer
compteur2=(i+1)*sizeLayer
listLayer.append(SGProperty[compteur1:compteur2])

listFinalLayer=[]

for z in range(nz):
layer=[]
for j in range(ny):
for i in range(nx):
k=j*nx+i
if listLayer[z][k]==0:
layer.append(sgems.nan())
else:
layer.append(listLayer[z][k])
listFinalLayer.append(layer)

print ("getListe3D Done")
return listFinalLayer;


def remplaceLayer(listeIndice,layerTrend,listeLayer,gridName,listeAssignation):
#listeIndice comporte les position des couches à remplacer
```

```
#layerTrend comporte la liste de couches trends
#listeLayer comporte la liste des couche du grid
#listeAssignation contient une liste de liste,
#la première liste contient les numéro des couches à modifier par le premier trend...

import sgems
grid_geom = sgems.get_grid_info(gridName)
nx = int(grid_geom['num_cells'][0]) #nombre de cellule en x
ny = int(grid_geom['num_cells'][1]) #nombre de cellule en y
nz = int(grid_geom['num_cells'][2]) #nombre de cellule en z

# for i in range(len(listeAssignation)):
# listeLayer[listeIndice[i]]=layerTrend[i]
zModif=[]
for i in range(len(layerTrend)):
for j in range(len(listeAssignation[i])):
listeLayer[listeAssignation[i][j]]=layerTrend[i]
zModif.append(listeAssignation[i][j])

gridBack = []

for z in range(nz):
for j in range(ny):
for i in range(nx):
k=j*nx+i
gridBack.append(listeLayer[z][k])


sgems.set_property('SG3D', 'Trend',gridBack )

print ("remplaceLayer Done")
return;

print "THE END"
```

A.5 DeeSse Parameters

```
/* SIMULATION GRID (SG) */
407    504     250               // size    in each direction
  100   100   2                 // spacing in each direction
  618986.25    1719964.75   -220  // origin

/* SIMULATION VARIABLES */
2
facies 1  DEFAULT_FORMAT //Doit correspondre au nom de la variable dans le fichier
trend 0

/* OUTPUT SETTINGS FOR SIMULATION */
OUTPUT_SIM_ONE_FILE_PER_REALIZATION
simuMPS100

/* OUTPUT REPORT */
/* Flag (0 / 1), and if 1, output report file. */
1
test_report.txt

/* TRAINING IMAGE */
1
tiSmall.gslib

/* DATA IMAGE FILE FOR SG */
1
trend.gslib

/* DATA POINT SET FILE FOR SG */
1
well.gslib

/* MASK IMAGE */
1
gridMask.gslib

/* HOMOTHETY */
*/
0

/* ROTATION */
2
1 orientation.gslib
0 0.0 0.0
0 0.0 0.0


/* CONSISTENCY OF CONDITIONING DATA (TOLERANCE RELATIVELY TO THE RANGE OF TRAINING VALUES) */
0.05

/* NORMALIZATION TYPE (FOR VARIABLES FOR WHICH DISTANCE TYPE IS NOT 0 AND DISTANCE IS ABSOLUTE) */
NORMALIZING_LINEAR

/* SEARCH NEIGHBORHOOD PARAMETERS */
/* SEARCH NEIGHBORHOOD PARAMETERS FOR VARIABLE #0 */
```

```
  20    20    0.0   // search radius in each direction
  1.0    1.0    1.0   // anisotropy ratio in each direction
  0.0    0.0    0.0   // angles (azimuth, dip, plunge in degrees) for rotation
  0.0                 // power for computing weight according to distance
/* SEARCH NEIGHBORHOOD PARAMETERS FOR VARIABLE #1 */
  20    20    0.0   // search radius in each direction
  1.0    1.0    1.0   // anisotropy ratio in each direction
  0.0    0.0    0.0   // angles (azimuth, dip, plunge in degrees) for rotation
  0.0                 // power for computing weight according to distance

/* MAXIMAL NUMBER OF NEIGHBORING NODES FOR EACH VARIABLE (as many number(s) as number of variable(s)) */
24
10

/* MAXIMAL DENSITY OF NEIGHBORING NODES IN SEARCH NEIGHBORHOOD FOR EACH VARIABLE (as many number(s)
   as number of variable(s)) */
1.0
1.0

/* RELATIVE DISTANCE FLAG FOR EACH VARIABLE (as many flag(s) (0 / 1) as number of variable(s)) */
0
0

/* DISTANCE TYPE FOR EACH VARIABLE (as many number(s) as number of variable(s)) */
/* Available distance (between data events):
      - 0: non-matching nodes (typically for categorical variable)
      - 1: L-1 distance
      - 2: L-2 distance
      - 3: L-p distance, requires the real positive parameter p
      - 4: L-infinity distance */
0
1

/* WEIGHT FACTOR FOR CONDITIONING DATA, FOR EACH VARIABLE (as many number(s) as number of variable(s)) */
5
0.5

/* SIMULATION AND PATH PARAMETERS */
PATH_RANDOM

/* DISTANCE THRESHOLD FOR EACH VARIABLE (as many number(s) as number of variable(s)) */
0.05
0.25

/* PROBABILITY CONSTRAINTS */
/* PROBABILITY CONSTRAINTS FOR VARIABLE #0 */
0
0

/* BLOCK DATA */
/* BLOCK DATA FOR VARIABLE #0 */
0
0

/* MAXIMAL SCAN FRACTION FOR EACH TI (as many number(s) as number of training image(s)) */
0.50

/* TOLERANCE */
```

```
/* Tolerance t on the threshold value for flagging nodes */
0.0

/* POST-PROCESSING */
1
POST_PROCESSING_PARAMETERS_DEFAULT

/* PYRAMIDS */
0

/* SEED NUMBER AND SEED INCREMENT */
1993
1

/* NUMBER OF REALIZATIONS */
100

END
```

A.6 Shannon Entropy Calculation

```
# -*- coding: utf-8 -*-
import sgems
import math

####################
#Valentin Juillet 2018
####################

grid_name     = "simu100_1"
grid_geom     = sgems.get_grid_info(grid_name)

allProba = []

for i in range(6):
propertyValu = sgems.get_property(grid_name,"PostSim100 Category "+str(i))
allProba.append(propertyValu)


nx = int(grid_geom['num_cells'][0]) #Nombre de cellule en x
ny = int(grid_geom['num_cells'][1]) #Nombre de cellule en y
nz = int(grid_geom['num_cells'][2]) #Nombre de cellule en z

listeProba=[]
proba=[]

for i in range(nx*ny*nz):

if str(propertyValu[i])!= "nan":

for j in range(6):
if allProba[j][i] != 0 :

proba.append(round(allProba[j][i],3))

listeProba.append(proba)
proba=[]

else:
listeProba.append(sgems.nan())

shannonList = []
probaLogList = []

for i in range(nx*ny*nz):

if str(listeProba[i])!="nan":
#Calcul Shannon

for j in range(len(listeProba[i])):

if listeProba[i][j] > 0.0001 :
probaLogList.append(listeProba[i][j]*(math.log(1/listeProba[i][j],5)))


entropie = sum(probaLogList)
```

```
shannonList.append(entropie)
probaLogList = []

else:
shannonList.append(sgems.nan())

print len(shannonList)
sgems.set_property(grid_name,"Shannon Entropy",shannonList)

print "THE END"
```

A.7 VPCs Tests

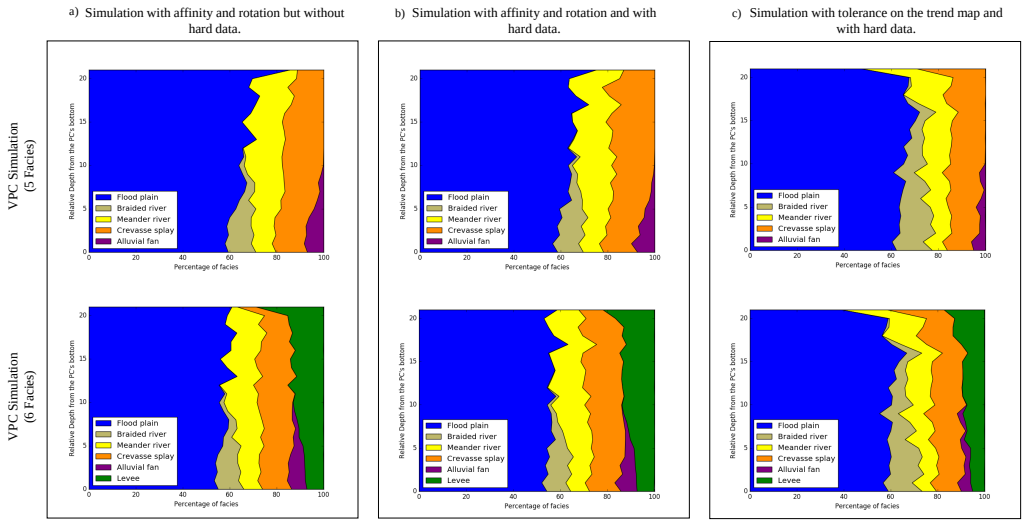


Figure 22 – Vertical proportion curves calculated from different simulations. The VPCs allow to visualize the differences between the simulations and validate the vertical proportion trend of a model.

A.8 Transmissivity Inversion/Optimization

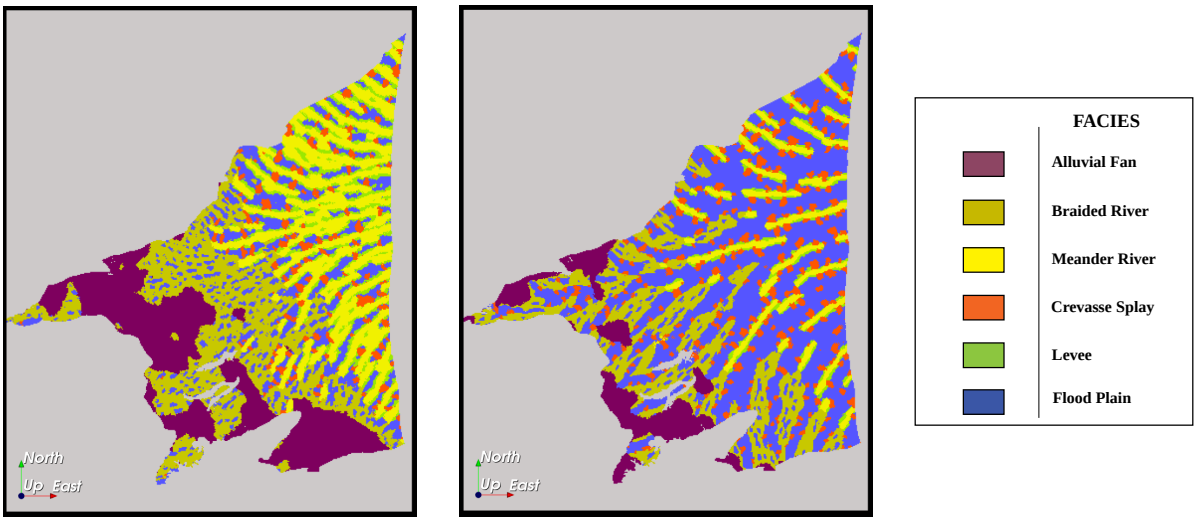
```
# -*- coding: utf-8 -*-
import numpy as np
import math
import pandas as pd
import matplotlib.pyplot as mp

#########################
#Valentin Juillet 2018
#########################

###
#On importe les données de facies tiré des simulations
#Le fichier csv se compose d'une liste de tout les facies présent dans la simulation,
#pour l'ensemble des points ou les données de transmissivité sont connues
###

directory1 = "C:\\Users\\wdall\\Desktop\\Python\\Code_OK\\SGEMs\\transmissivityMap\\KfromSimu.csv"
dataK      = pd.read_csv(directory1)
KforT      = []

for i in range(188):
   KforT.append(list((dataK["T"+str(i)])))

#On crée les variables que l'on va compter ainsi que la liste vide stockant ces variables
compteNonNulle  = 0
compteZero      = 0
compteUn        = 0
compteDeux      = 0
compteTrois     = 0
compteQuatre    = 0
compteCinq      = 0
compteNonNulle  = 0
indiceAll       = []

for j in range(len(KforT)):
    indiceK = []
    compteNonNulle = 0
    compteZero     = 0
    compteUn       = 0
    compteDeux     = 0
    compteTrois    = 0
    compteQuatre   = 0
    compteCinq     = 0

    for i in range(len(KforT[j])):

        if KforT[j][i]   == 0:
            compteZero     += 1
            compteNonNulle += 1

        elif KforT[j][i] == 1:
            compteUn       += 1
            compteNonNulle += 1
```

```
        elif KforT[j][i] == 2:
            compteDeux     += 1
            compteNonNulle += 1

        elif KforT[j][i] == 3:
            compteTrois    += 1
            compteNonNulle += 1

        elif KforT[j][i] == 4:
            compteQuatre   += 1
            compteNonNulle += 1

        elif KforT[j][i] == 5:
            compteCinq     += 1
            compteNonNulle += 1

    indiceK.append(compteZero)
    indiceK.append(compteUn)
    indiceK.append(compteDeux)
    indiceK.append(compteTrois)
    indiceK.append(compteQuatre)
    indiceK.append(compteCinq)
    indiceK.append(compteNonNulle)

    indiceAll.append(indiceK)


###
#On récupère les valeurs de Transmissivité mesurées/réelles
#ATTENTION il faut diviser les valeurs de Transmissivité par 1000
###

directory2     = "C:\\Users\\wdall\\Desktop\\Python\\Code_OK\\SGEMs\\transmissivityMap\\Tvalue.csv"
dataT          = pd.read_csv(directory2)
Td             = dataT["T"]
Transmissivity = []

###
#On créé une liste de Transmissivité
#On transforme les valeurs de transmissivité en une matrice (188,1)
###

for i in range(len(dataT)):
    Transmissivity.append(Td[i]/(1000))

Tmatrix=np.zeros((len(Transmissivity),1))

for i in range(len(Transmissivity)):
    Tmatrix[i,0] = Transmissivity[i]

###
#On crée une matrice correspondant à l'épaisseur de chaque facies pour chaque point de transmissivité
###

IndiceMatrix = np.zeros((len(indiceAll),6))

for i in range(len(indiceAll)):
```

```
     for j in range(6):
         IndiceMatrix[i,j] = indiceAll[i][j]

IndiceMatrix = IndiceMatrix*2                #

###
#Calcul matriciel 1
###

A = IndiceMatrix.transpose()
B = np.dot(A,IndiceMatrix)
C = np.linalg.inv(B)
D = np.dot(C,A)

Kmatrix=np.dot(D,Tmatrix)

###
#Vérification calcul matriciel 1
###

T_theo_1 = np.dot(IndiceMatrix,Kmatrix)
mp.loglog(T_theo_1,Tmatrix,"+", label= "Matrix Calculation Method")
mp.show()


###
#Calcul matriciel 2
#On cherche cette fois à calculer les valeurs de K par optimisation
#On va ainsi pouvoir définir des bornes aux valeurs de K (K doit être positif)
#On va également chercher à minimiser l'erreur Ki carrée entre les valeurs de
#Transmissivité mesurées et calculées
#Ki carré = [SOMME (log10(T_mesurées)-log10(T_calculées))]**2
###

from scipy.optimize import minimize

#1 On definie la fonction dont les paramètres sont à optimiser en vue de la minimisation
#On veut minimiser le Ki log en ajustant les paramètres K

def fToMinimize(Kin):

    K = np.zeros((6, 1))
    for j in range(6):
        K[j, 0] = Kin[j]


    #Le K input doit être de forme matricielle (6,1)
    T_theo_2        = np.dot(IndiceMatrix,K)
    goodList        = (T_theo_2!=0) & (Tmatrix!=0)
    Tcal_trie       = T_theo_2[goodList]
    Tmesu_trie      = Tmatrix[goodList]
    Tcal_log        = np.log10(Tcal_trie)
    Tmesu_log       = np.log10(Tmesu_trie)

    Ki_carre = np.dot((Tmesu_log-Tcal_log),(Tmesu_log-Tcal_log))

    return Ki_carre;
```

```
#2 On definie l'objective fonction

def objective(K):
return fToMinimize(K);

#3 On définie les boundaries des variables

bnds0 = ((0,None),(0,None),(0,None),(0,None),(0,None),(0,None))

#4 On définie les valeurs K0
K0       = [0.005,0.2,0.25,0.01,0.05,0.01]  # fonction = 43   ; 310


#5 On définie le problème à optimiser par l'algorithme
sol = minimize(fToMinimize,K0,method="SLSQP",bounds=bnds0,options={"disp":True})

KOptimal = sol.x

#6 On re-calcul la valeur de T pour les plotées par rapport à notre permière estimation des variables K
KoptM = np.reshape(KOptimal,(6,1))
TCalcOpti=np.dot(IndiceMatrix,KoptM)

mp.loglog(TCalcOpti,Tmatrix,"r.",label = "Optimization Method")
mp.loglog([1e-5,1e-2],[1e-5,1e-2],"k")
mp.xlabel("T Calculed")
mp.ylabel("T Mesured")
mp.legend()
mp.show()

print "THE END"
```

A.9 New Simulations test



a) Without lateral repetition of the TI

b) Without Hard Data

Figure 23 – Bottom layer of the two 3D simulation tests realized in order to understand the meander beds spatial variation. (a) The first set of simulations was realized without a lateral repetition in the TI. The propability maps of this set (Annexe 10) were more heterogeneous than the set presented in the result section (4.1) . However, the floodplain proportion was underestimated and the lateral connectivity of the beds was too important. (b) The second set was realized with the first TI (Figure 11), which displayed lateral repetitions but without hard data . The simulation appeared to be more realistic than the one without repetitions in the TI (a). However, its probability maps showed the same homogeneous distribution than the one with hard data and lateral repetition (4.2).

A.10 Probability Maps for the new sets of Simulations

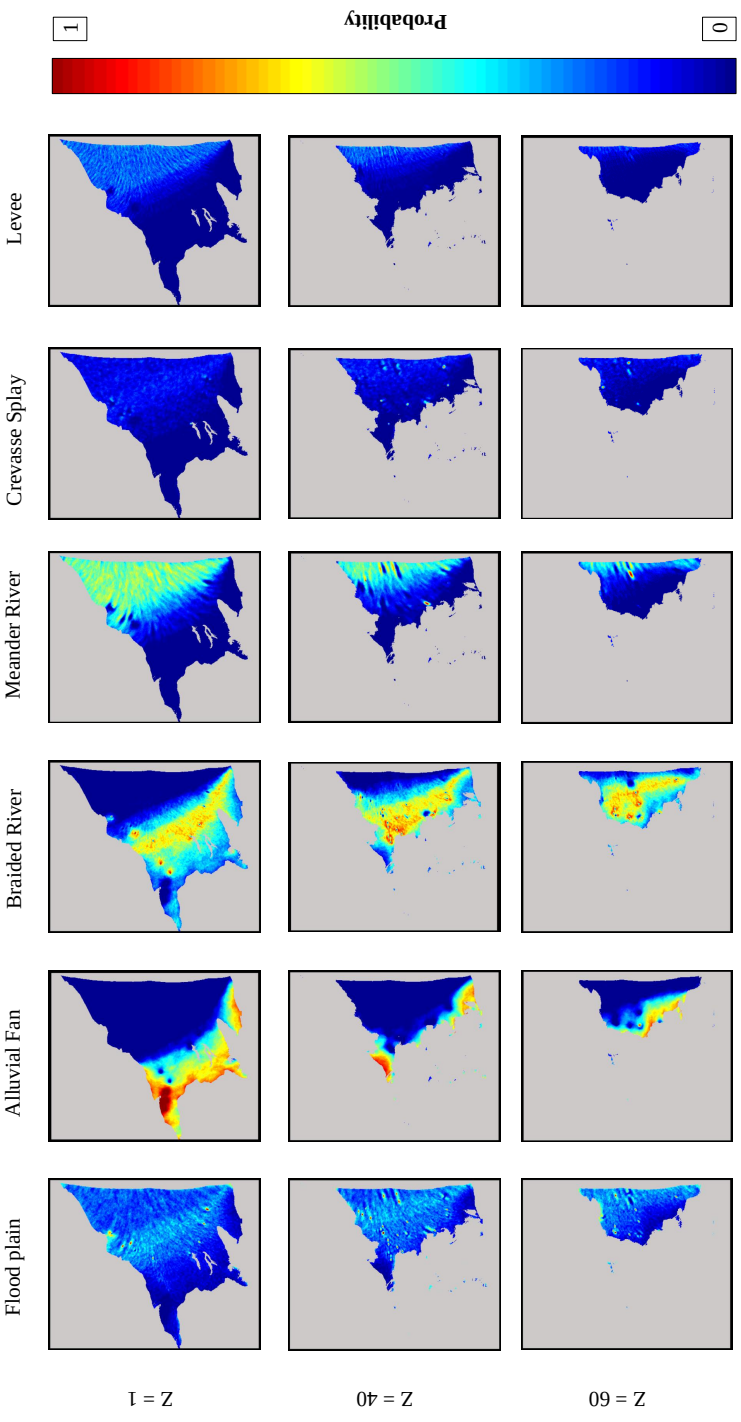


Figure 24 – Probability maps for a set of 100 simulations. The parameters used were the same as the one used in the previous simulations (4.1). We used a modified TI that did not show any lateral repetition of its patterns. The meander facies display a higher variation in its location over the plain. It appears that the distance between every channel could have been too defined in the previous TI (Figure 11 e).

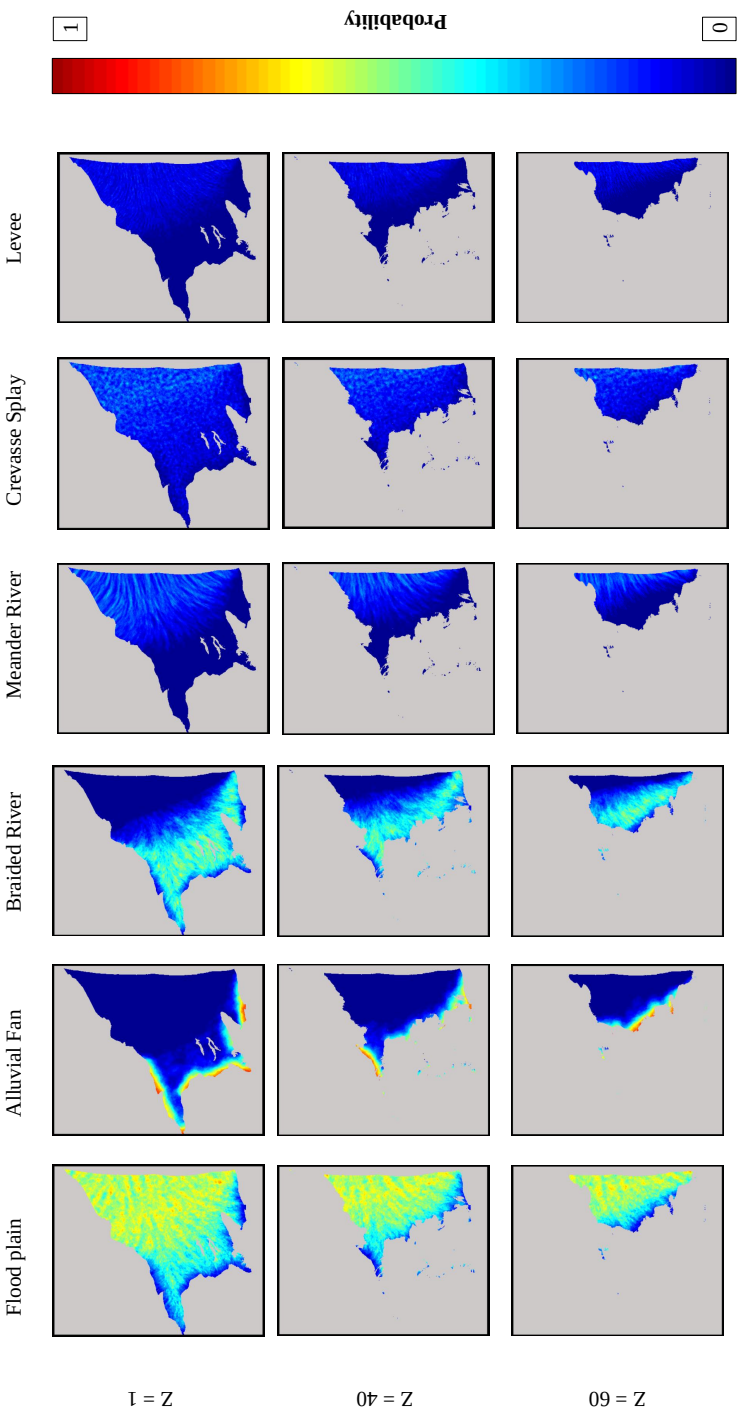


Figure 25 – Probability maps for a set of 100 simulations. The parameters used were the same as the one use in the previous simulations (4.1). We did not constrained these simulations with hard data. Even without hard data, the distance between the meander beds appears to be too constrained and it does not show variability in the central zone of the plain.

A.11 Training Image Workflow

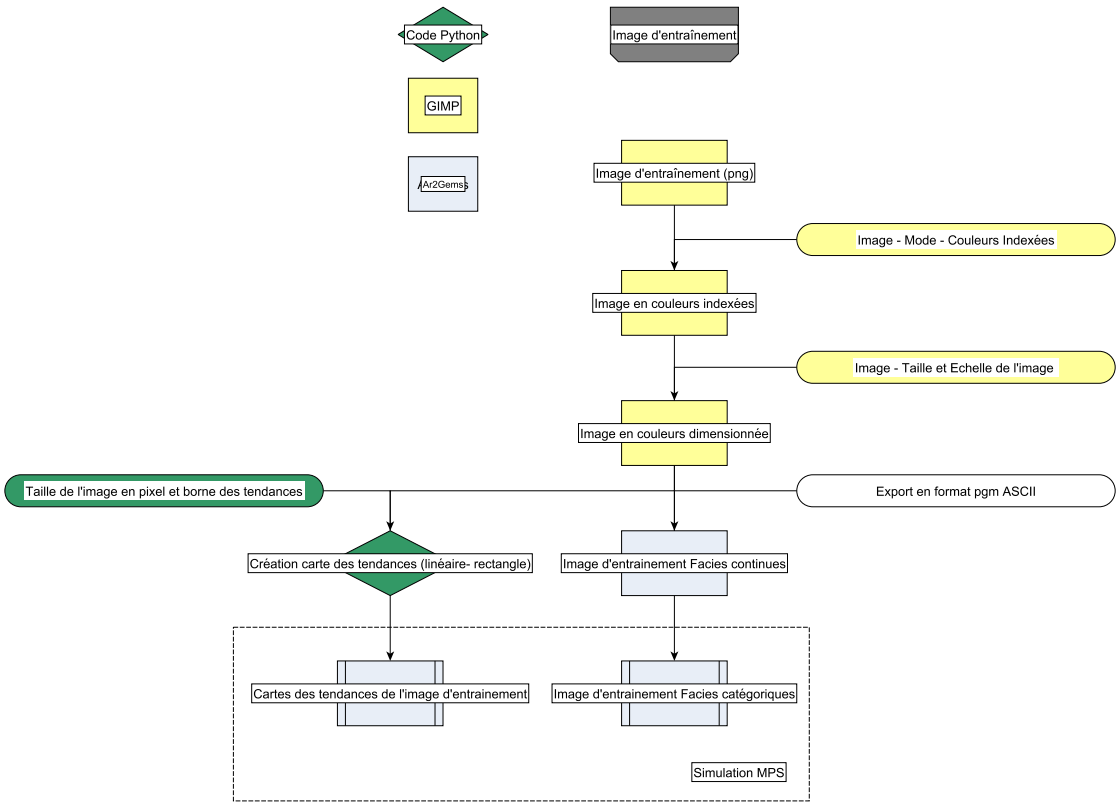


Figure 26 – Chart of the main steps for the creation of a TI.

A.12 Trend maps Workflow

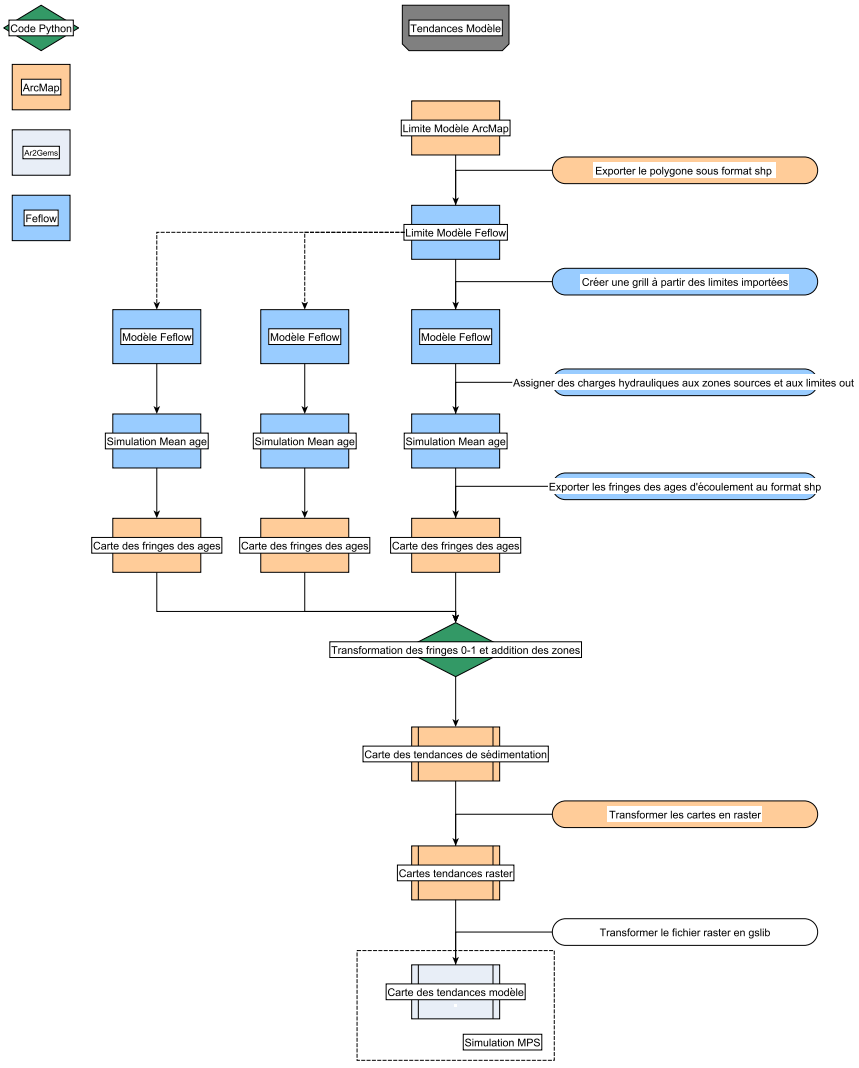


Figure 27 – Chart of the main steps for the creation of a complex trend map.

A.13 Rotation map Workflow

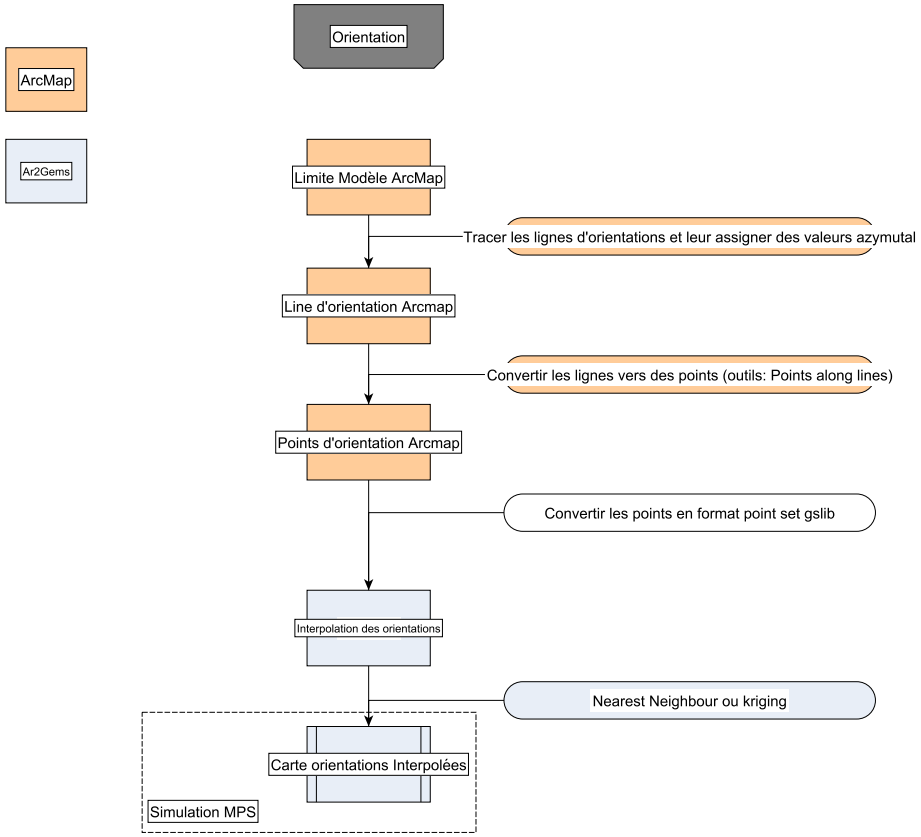


Figure 28 – Chart of the main steps for the creation of a continuous rotation map.

References

AR2GEMS (2010). Ar2tech products ar2gems : A modern geostatistical platform. http://www.ar2tech.com.

ArcMap (2016). Esri geographic information system company, arcmap 10.5. https://www.esri.com.

Caballero, Y. and Ladouche, B. (2015). Impact of climate change on groundwater in a confined Mediterranean aquifer. *Hydrology and Earth System Sciences Discussions*, 12(10):10109–10156.

Cazenave, A. (2013). Hausse du niveau de la mer et impact du changement climatique global. *La lettre du Collège de France*.

Chauveau, M., Chazot, S., Perrin, C., Bourgin, P.-Y., Sauquet, E., Vidal, J.-P., Rouchy, N., Martin, E., David, J., Norotte, T., Maugis, P., and De Lacaze, X. (2013). Quels impacts des changements climatiques sur les eaux de surface en France à l'horizon 2070 ? *La Houille Blanche*, (4):5–15.

CLE (2011). Sage des nappes plio-quaternaire de la plaine du roussillon. Technical report, Commission Locale de l'Eau.

Comunian, A., Renard, P., Straubhaar, J., and Bayer, P. (2011). Three-dimensional high resolution fluvio-glacial aquifer analog - Part 2: Geostatistical modeling. *Journal of Hydrology*, 405(1-2):10–23.

de Carvalho, P. R. M., da Costa, J. F. C. L., Rasera, L. G., and Varella, L. E. S. (2017). Geostatistical facies simulation with geometric patterns of a petroleum reservoir. *Stochastic Environmental Research and Risk Assessment*, 31(7):1805–1822.

Dörfliger, N. and Perrin, J. (2012). Ressource en eau : une gestion nécessairement locale dans une approche globale.

Dutartre, P., Desprats, J., and Rouzeau, . (1995). Prospection hydrogeologique des milieux fissurés du socle, Massif des Albères - France. Technical report, BRGM.

Duvail, C. (2007). Expression des facteurs régionaux et locaux dans l'enregistrement sédimentaire d'une marge passive - Exemple de la marge du Golfe du Lion étudiée selon un continuum terre-mer. *Universite Montpellier 2, Sciences et Techniques du Languedoc*.

FEFLOW (2016). Mike powered by dhi, feflow 7.0. https://www.mikepoweredbydhi.com.

Hu, L. Y. and Chugunova, T. (2008). Multiple-point geostatistics for modeling subsurface heterogeneity: A comprehensive review. *Water Resources Research*, 44(11):1–14.

Lofi, J., Gorini, C., Berné, S., Clauzon, G., Reis, A. T. D., Ryan, W. B., and Steckler, M. S. (2005). Erosional processes and paleo-environmental changes in the western gulf of lions (sw france) during the messinian salinity crisis. *Marine Geology*, 217(1):1 – 30.

Mariethoz, G., Renard, P., and Straubhaar, J. (2010). The direct sampling method to perform multiple-point geostatistical simulations. *Water Resources Research*, 46(11):1–14.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, 58(8):1246–1266.

Meerschman, E., Pirot, G., Mariethoz, G., Straubhaar, J., Van Meirvenne, M., and Renard, P. (2013). A practical guide to performing multiple-point statistical simulations with the Direct Sampling algorithm. *Computers and Geosciences*, 52:307–324.

Nichols, G. (2009). *Sedimentology & Stratigraphy*. A John wiley and Sons, LTD, Publication.

Petrel (2017). Schlumberger, petrel e&p software platform. https://www.software.slb.com.

Ravenne C, Galli A, D. B. B. H. (2002). Quantification of Facies Relationships Via Proportion Curves. *Geostatistics Rio 2000*, Quantitati(12).

Renard, P., Alcolea, A., and Ginsbourger, D. (2013). Stochastic versus Deterministic Approaches. pages 133–149.

Serra, O., Sulpice, L., et al. (1975). *Sedimentological analysis of shale-sand series from well logs*.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 5(3):3.

Straubhaar, J. (2017). User 's Guide May 2017 DeeSse Software patented by the University of Neuchâtel. (May).

Straubhaar, J., Renard, P., Mariethoz, G., Froidevaux, R., and Besson, O. (2011). An improved parallel multiple-point algorithm using a list approach. *Mathematical Geosciences*, 43(3):305–328.

Strebelle, S. (2002). Conditional simulation of complex geological structures using multiple-point statistics. *Mathematical Geology*, 34(1):1–21.

Strebelle, S., Payrazyan, K., and Caers, J. (2002). Modeling of a Deepwater Turbidite Reservoir Conditional to Seismic Data Using Multiple-Point Geostatistics. *SPE Annual Technical Conference and Exhibition*.

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