https://doi.org/10.35662/unine-thesis-2959

University of Neuchâtel, Switzerland

Faculty of Sciences

Centre for Hydrogeology and Geothermics (CHYN)

Discrete stochastic inversion: getting closer to hydrogeological applications

Thesis presented for the degree of

Docteur ès sciences

by

Przemysław Juda

Thesis advisor:	Prof. Philippe Renard
Rapporteurs:	Prof. Niklas Linde
	Prof. Thomas Mejer Hansen
	Dr. Jeremy White
Examinator:	Julien Straubhaar

Defended on 31 January 2022

UNIVERSITÉ DE NEUCHÂTEL

Faculté des Sciences
Secrétariat-décanat de Faculté
Rue Emile-Argand 11
2000 Neuchâtel – Suisse
Tél : + 41 (0)32 718 21 00
E-mail : secretariat.sciences@unine.ch

IMPRIMATUR POUR THESE DE DOCTORAT

La Faculté des sciences de l'Université de Neuchâtel autorise l'impression de la présente thèse soutenue par

Monsieur Przemysław JUDA

Titre:

"Discrete stochastic inversion: getting closer to hydrogeological applications"

sur le rapport des membres du jury composé comme suit:

- Prof. Philippe Renard, Université de Neuchâtel, Suisse
- Prof. Niklas Linde, Université de Lausanne, Suisse
- Assoc. Prof. Thomas Mejer Hansen, Aarhus University, Denmark
- Dr Jeremy White, INTERA, Austin, USA
- Dr Julien Straubhaar, Université de Neuchâtel, Suisse

Neuchâtel, le 14 mars 2022

Le Doyen, Prof. A. Bangerter

Acknowledgements

My thesis advisor, Philippe Renard, proposed me to embark on the PhD journey four years ago and I am very grateful to him for doing so. He has been a lovely person to work with — not only as an excellent academic, but also, and more importantly, as a warm, open, honest and patient mentor. If not Philippe, I would not have accomplished this work.

I also appreciate that other people have accompanied me during this endeavor, especially the colleagues from my team — Stochastic Hydrogeology and Geostatistics group.

Julien Straubhaar has always been ready to share his expertise in multiple-point statistics, mathematics and algorithmic details of geostatistical methods. His sharp eye for details saved many small (and bigger) errors in this thesis and in manuscripts submitted to journals.

Christoph Jäggli had the initial idea for speeding up posterior population expansion (PoPEx) with machine learning, and his clear explanations of workings of PoPEx algorithm allowed me to progress quickly on the relevant parts of the thesis. Christoph was also my guide in the academic universe during my first year of the PhD.

Yasin Dagasan offerred me a lot of advice on machine learning, when I was working on classifiers for speeding up PoPEx. He also implemented some early ideas and suggested useful software tools.

Dan Thuy-Lam has always been a kind listener, and we have had many discussions about discrete stochastic inversion. We exchanged clarifications and remarks. Dan answered eagerly my questions about her ensemble smoother with multiple data assimilation, which facilitated the implementation of the method for the inversion comparison study.

Valentin Dall'Alba-Arnau provided me with the Roussillon data, and thus enabled me to show a realistic use case of the cross-validation framework. He and Alexis Neven gave me Tsanfleuron data which I used for testing of Direct

Sampling Best Candidate.

I am also thankful for having participated in projects of other people, they have often provided me with the inspiration for the core of my thesis. Manon Trottet's project on ERT inversion inspired me to develop the tempered likelihood. Enrico Scarpa brought data of HES site in Poitiers for cross-validation, and it revealed limitations of the framework and showed perspectives for further developments. Augustin Gouy shared his synthetic inversion set-up, which served as a basis to construct the set-up used in the inversion comparison part.

The defense of the thesis was a special moment of an in-depth discussion about this work, and I would like to thank in particular the external jury members: Thomas Mejer Hansen, Niklas Linde, Jeremy White. They took time to review the thesis, shared their point of view and insightful remarks.

Many thanks to my colleagues and administrative personnel of the Centre of Hydrogeology and Geothermics, who cared for a pleasant and productive workplace. Last but not least, my family, especially Anna and Barbara, have showed enthusiasm and interest towards my doctorate. Thank you for your love and unconditional support.

Abstract

Stochastic discrete inversion methods allow capturing geological realism and quantify uncertainty, the two aspects that are crucial for groundwater management and the application of stochastic methods in hydrogeology. However, these methods present two major practical challenges: the choice of a correct prior representation and a high computational cost. This thesis addresses these challenges to facilitate future applications of discrete stochastic inversion on hydrogeological data.

Strategies for prior selection in the context of geostatistical simulations, and in particular multiple-point statistics are presented. When prior conditioning data is available, a cross-validation framework for categorical variables can be used with scoring rules. The framework can be used for tuning every parameter of geostatistical simulations, for example, choosing the training image for multiple point-statistics. A test case representing a simplified model of the Roussillon plain aquifer confirms the validity of the framework. Another tool presented in this thesis is the Direct Sampling Best Candidate (DSBC) algorithm, which has fewer algorithmic features than the Direct Sampling (DS) algorithm. It retains, however, all the advantages of DS, but simplifies the choice of the parameters, which is often done before the inversion. For the test cases that we studied, the simulation quality of DSBC was better than DS for conditional simulations, and slightly worse, but satisfactory, for unconditional simulations.

As for improving the computational performance of the inversion, machine learning algorithms are proposed to speed-up posterior population expansion (PoPEx). With random forest and AdaBoost, speed-up factors of PoPEx of around two times were observed, when applied to a synthetic tracer test data. These machine learning techniques have the potential to be used for other Monte Carlo inversions. Another solution for improving PoPEx convergence was also developed: a tempered likelihood, allowing to improve the uncertainty quantification. It alleviates the need to reduce the dimensionality of the data before inversion (as suggested by previous studies on PoPEx) and mitigates the problem of a very sharp likelihood function. The final point of the thesis is a comparison of three recent discrete inversion methods: PoPEx, ensemble smoother with multiple data assimilation, and DREAM-ZS. A synthetic but

realistic pumping test case showed that all three methods perform fairly well, provided that a correct prior is used. However, the choice of the prior is essential, and with wrong priors, represented by different training images, the performance of the methods is strongly affected. The performance was measured with probabilistic scores on assimilated data and the 10-day groundwater protection zone.

Keywords

geostatistics; inverse problem; uncertainty quantification; stochastic inversion; posterior population expansion; Monte Carlo sampling; data assimilation; ensemble smoother; machine learning; binary classification; deep learning; multiple-point statistics; cross-validation; groundwater flow and transport; categorical fields

Résumé

Les méthodes d'inversion discrète stochastiques permettent de reproduire correctement la situation géologique et de quantifier l'incertitude. Ce deux aspects sont cruciaux pour la gestion des eaux souterraines et pour l'application des méthodes stochastiques en hydrogéologie. Cependant, dans la pratique ces méthodes présentent deux défis majeurs : le choix d'une représentation *a priori* correcte et un coût de calcul élevé. Cette thèse aborde ces problèmes afin de faciliter les applications futures de l'inversion stochastique discrète sur les données hydrogéologiques.

Des stratégies sont présentées pour la sélection de la représentation *a priori* dans le contexte des simulations géostatistiques, et en particulier des simulations multipoints. Lorsque des données de conditionnement sont disponibles, une méthode de validation croisée pour les variables catégorielles peut être utilisé. Cette méthode permet de régler n'importe quel paramètre des simulations géostatistiques, par exemple le choix de l'image d'entraînement pour les simulations multipoints. Un cas test avec un modèle simplifié de l'aquifère de la plaine du Roussillon a confirmé la validité de la méthode. Un autre outil présenté dans cette thèse est l'algorithme DSBC (Direct Sampling Best Candidate), qui possède moins de paramètres algorithmiques que l'algorithme DS (Direct Sampling). Il conserve néanmoins tous les avantages de DS, mais simplifie le choix des paramètres, qui est souvent effectué avant l'inversion. Pour les cas tests que nous avons étudiés, la qualité de simulation de DSBC était meilleure que celle de DS pour les simulations conditionnelles, et légèrement moins bonne, mais satisfaisante, pour les simulations non conditionnelles.

Quant à l'amélioration des performances computationnelles de l'inversion, des algorithmes d'apprentissage automatique sont proposés pour accélérer l'algorithme PoPEx (posterior population expansion). Avec le Random Forest et AdaBoost, des facteurs d'accélération de PoPEx d'environ deux fois ont été observés, lorsqu'ils ont été appliqués à un cas synthétique d'inversion des données d'essai de traçage. Ces techniques pourraient être utilisées pour d'autres algorithmes d'inversion Monte Carlo. Une autre solution pour améliorer la convergence (et la quantification de l'incertitude) PoPEx a également été développée : la vraisemblance tempérée (*tempered likelihood*). Elle permet

d'éviter de réduire la dimensionnalité des données avant l'inversion (comme suggéré par les études précédentes sur PoPEx) et atténue le problème d'une fonction de vraisemblance très pointue. Le point final de la thèse est une comparaison de trois méthodes récentes d'inversion discrète : PoPEx, *ensemble smoother with multiple data assimilation* (ESMDA), et DREAM-ZS. Un cas avec les données synthétiques (mais réalistes) d'un test de pompage a montré que les trois méthodes sont assez performantes, à condition d'utiliser la représentation du *prior* correcte. Cependant, le choix du *prior* est essentiel, et avec des représentations mauvaises, représentées par différentes images d'entraînement, les performances des méthodes sont fortement affectées. Les performances ont été mesurées à l'aide de scores probabilistes sur des données assimilées et sur la zone de protection des eaux souterraines de 10 jours.

Mots-clés

géostatistique ; problème inverse ; quantification de l'incertitude ; inversion stochastique ; posterior population expansion ; méthodes de Monte Carlo ; assimilation de données ; ensemble smoother; apprentissage automatique ; classification binaire ; réseaux neuronaux ; statistique multipoints ; validation croisée ; écoulement des eaux souterraines et transport ; champs catégoriels

Contents

Chapter 1

Introduction

With climate change and growing population, more and more stress is put on groundwater resources, and in many places, it is crucial to get a better understanding of groundwater flow. One tool that is used in this perspective is numerical modeling. But these models need the input of subsurface parameters (hydraulic conductivity, specific storage, porosity, etc.), which govern the flow and transport properties. Direct measurements of these parameters are possible, but usually result in very sparse observations. It is much more convenient to measure state variables by monitoring groundwater levels or quality, performing tracer test, pumping test, or slug test. For any given subsurface parameters and experiments, the state variables can be simulated by solving the so-called forward problem. In the case of groundwater flow, it might be for example Darcy equation or advection diffusion reaction equation for contaminant transport. These are usually partial differential equations, and can be computationally expensive to solve, but well-studied numerical schemes are available. The forward problem links a specific set of subsurface parameters, let us call it a model — with the observations. The mapping is unique and well-defined — for a given model, the solution (the observations) is unique. The inverse problem lies in solving it the other way round: determining the subsurface parameters from the observations.

Why stochastic inverse problem?

Unfortunately, deterministic inverse problems are usually ill-posed, their solutions are non-unique (and often unstable). The stochastic framework allows setting the problem so that its solution becomes well-defined and unique. However, it comes at a price: the solution is a probability distribution over a non-linear high-dimensional space (manifold). Solving such a problem usually requires expensive Monte Carlo methods and many forward model calls. There are also benefits of using this framework. The uncertainty of the solution can be quantified and expressed in rather intuitive terms. The uncertainty quantification help to make better, more conscious, management decisions concerning

groundwater resources.

Why discrete problem?

The subsurface is highly heterogeneous because it has been shaped by geological processes at different temporal and spatial scales. The heterogeneity often involves different and discrete rock types (geological facies), having distinct petrophysical properties. Therefore, modeling the geology with categorical variables captures better the reality than assuming a continuous distribution. It is also essential for certain geological features, such as karstic cavities or fractures, which cannot be reasonably described with continuous variables. Moreover, connectivity properties, essential in describing the transport in groundwater, are often better represented with the categorical approach. Lam (2019) showed that solving inverse problem with continuous variables, breaks connectivity patterns. Sometimes, it might be a viable compromise (due to much smaller computational cost), but we will limit our scope to highly heterogenous problems.

Bottlenecks of discrete stochastic inversion

We have already mentioned that the computational cost is one of the main bottlenecks of stochastic inversion, this is obviously also the case for discrete stochastic inversion. Another major challenge is the identification of the prior distribution, which in real case applications is unknown. In this thesis, I am addressing these two main bottlenecks central to the stochastic inversion of discrete spatial random fields.

In the first part, I will focus on the prior — its choice, which includes the choice of the geostatistical method used to model the heterogeneous categorical fields. We will also look at the parameter choice for geostatistical algorihtms. The second part deals with the core of the inversion algorithms: to speed-up and improve uncertainty quantification of the inversion, and then comparing performance of different inversion frameworks. The comparison will bring us back to the problem of the prior choice, because we will look at the performance and convergence of the inversion with different priors.

The main objective of this work is therefore to provide some new knowledge and tools to facilitate the application of stochastic discrete inversion for complex cases with real measurements.

Structure of the thesis

In Chapter 2 (published as an article[1]), a cross-validation methodology for categorical geostatistical simulations is presented. It can be applied when cat-

[1] Juda, P., Renard, P., & Straubhaar, J. (2020). A framework for the cross-validation of categorical geostatistical simulations. *Earth and Space Science*, 7, e2020EA001152. `https://doi.org/10.1029/2020EA001152`

egorical data about the subsurface are available. For example, when lithologies are known at certain locations (borehole data), and one wants to determine the best stochastic simulation method for such a set-up. In the context of the inverse problem, it is a tool that is designed to assist the selection of a prior (the geostatistical algorithm, the training image, the parameter), when discrete conditioning data are available.

Chapter 3 presents a simplified version of the Direct Sampling (DS) algorithm: Direct Sampling Best Candidate (DSBC). The new algorithm — or the new way to choose DS parameters — addresses the difficulty of choosing the optimal simulation parameters. DSBC reduces the number of computational parameters with respect to DS. In this way, users can more quickly tune the geostatistical method prior to running inversion.

Chapter 4 (published as an article[2]) addresses the problem of the computational cost of the Monte Carlo samplers for hydrogeology. It presents an application of machine learning algorithms for speeding up Posterior Population Expansion algorithm (which is a Monte Carlo, adaptive importance sampler). The presented methodology allows accelerating the core of the algorithm, independently of the geostatistical method. Moreover, the methodology is quite generic and could potentially be used for other Monte Carlo algorithms (Metropolis-Hastings sampler, DREAM, etc.).

Chapter 5 presents a tempered likelihood for Posterior Population Expansion (PoPEx). The tempered likelihood aims to improve the uncertainty representation of PoPEx, in the cases where many observation data are available. These cases are difficult, as in practice most models suffer from negligible likelihood and there are too few models to form the prediction (posterior). The tempered likelihood is an approximation which allows including more models in the prediction by artificially inflating the likelihood term.

Chapter 6 compares different inversion algorithms and attempts to answer two important questions: what algorithm to choose for the inverse problem, and how important is the choice of the prior? In a real case scenario, the model space might not exactly contain the real field. What happens if we make a mistake in the prior choice? We also present probabilistic scores and a framework to compare the algorithms.

The thesis also contains an Appendix A with additional figures and tables related to chapters 3 and 6. These results were not included in the chapters for the sake of brevity.

Finally, during the thesis, I participated in the research that lead to two publi-

[2]Juda P and Renard P (2021) An Attempt to Boost Posterior Population Expansion Using Fast Machine Learning Algorithms. *Front. Artif. Intell.* 4:624629. doi:10.3389/frai.2021.624629

cations. Appendix B presents an application of generative adversarial network (GAN) as a fast forward solver, aiming to speed up the inversion. A test case of inverting steady-state hydraulic heads with PoPEx was developed to check how GAN forward emulator performs in this setting. My contributions involved integrating GAN into the inversion framework, running the inversion, quantifying its results, and writing the corresponding part of the text. The second publication is not included in the manuscript. It presents an application of multiple-point statistics to estimation of volume of Tsanfleuron glacier[3]. I participated in the data acquisition, designed and implemented the quality indicators and wrote the corresponding section. These quality indicators are used in Chapter 3, Test Case 2.

[3]Neven, A., Dall'Alba, V., Juda, P., Straubhaar, J., & Renard, P. (2021). Ice volume and basal topography estimation using geostatistical methods and ground-penetrating radar measurements: Application to the Tsanfleuron and Scex Rouge glaciers, Swiss Alps. The Cryosphere, 15(11), 5169–5186. https://doi.org/10.5194/tc-15-5169-2021

Chapter 2

A framework for the cross-validation of categorical geostatistical simulations[1]

Abstract

The mapping of subsurface parameters and the quantification of spatial uncertainty requires selecting adequate models and their parameters. Cross-validation techniques have been widely used for geostatistical model selection for continuous variables, but the situation is different for categorical variables. In these cases, cross-validation is seldom applied, and there is no clear consensus on which method to employ. Therefore, this paper proposes a systematic framework for the cross-validation of geostatistical simulations of categorical variables such as geological facies. The method is based on K-fold cross-validation combined with a proper scoring rule. It can be applied whenever an observation data set is available. At each cross-validation iteration, the training set becomes conditioning data for the tested geostatistical model, and the ensemble of simulations is compared to true values. The proposed framework is generic. Its application is illustrated with two examples using multiple-point statistics simulations. In the first test case, the aim is to identify a training image from a given data set. In the second test case, the aim is to identify the parameters in a situation including nonstationarity for a coastal alluvial aquifer in the south of France. Cross-validation scores are used as metrics of model performance and quadratic scoring rule, zero-one score, and balanced linear score are compared. The study shows that the proposed fivefold stratified cross-validation with the quadratic scoring rule allows ranking the geostatistical models and helps to identify the proper parameters.

[1]This chapter was published as: Juda, P., Renard, P., & Straubhaar, J. (2020). A framework for the cross-validation of categorical geostatistical simulations. *Earth and Space Science*, 7, e2020EA001152. `https://doi.org/10.1029/2020EA001152`

2.1 Introduction

When modeling heterogeneous media, the choice of a suitable geostatistical approach is often a challenge. In the case of categorical fields (e.g. geological facies), many approaches are available: such as sequential indicator simulations (SIS), T-Progs, truncated gaussian and pluri-gaussian simulations, multiple-point statistics (MPS), object-based models, genetic and pseudo genetic approaches (see e.g. Chiles and Delfiner 2012; Pyrcz and Deutsch 2014 for a broad presentation of the methods).

Moreover, results obtained by all estimation or simulation methods depend on the model and computational parameters in a complex manner. A simple and powerful tool for statistical model selection is cross-validation: it consists of removing some data and comparing them to the predictions generated by the model. The first rigorous treatment of cross-validation was developed by Stone (1974) and Geisser (1974) simultaneously. In the former work, the term "cross-validation" appeared for the first time and was presented as the leave-one-out technique, where one data point is left out at a time and compared to the prediction; the procedure is then repeated for all points. Geisser (1975) generalized the idea of cross-validation to leaving out several points at a time. This variant of cross-validation has been later called also "multifold" (for example by Zhang (1993)) and currently some refer to it as "v-fold" (see for example Arlot and Celisse (2010)), but the most commonly used term is "K-fold cross-validation" (Hastie et al. 2009). It consists in partitioning the data into K subsets, one subset used for testing at a time. Another simpler technique is random subsampling, also called hold-out sampling. In this method, a random subset of the data, the hold-out test set, is removed from the dataset. The model is trained on the remaining part and its performance measured with the test set.

Currently, K-fold cross-validation is the technique most often used for model selection in classification problems. One of the early examples includes the work of Breiman et al. (1984). Breiman and Spector (1992) showed that leave-one-out performs worse than 5-fold for model selection, which is comparable to bootstrap but less expensive computationally. Kohavi (1995) provided some more insight into different cross-validation techniques by comparing performance with the different number of folds, leave-one-out, and bootstrap methods. He found that 10-fold cross-validation is a better choice even if the leave-one-out technique is computationally feasible, with 5-fold found performing nearly as good as 10-fold. He also suggested the use of stratified cross-validation: each of the subsets should have roughly the same proportion of classes as the whole dataset. Rodriguez et al. (2010) performed a study of the sensitivity of K-fold and concluded that 5-fold or 10-fold should be used. The goal of cross-validation is to estimate prediction error on unknown data (Hastie et al. 2009). That is why it repeats the train-test split and averages the obtained errors: it strives to measure prediction error when using the whole available

dataset. For measuring the error, either a loss function is used, penalizing wrong predictions (the lower loss the better); or a scoring function, rewarding correct predictions (the higher the better). Typically, a loss function compares a single predicted value with a single true value and averages the mismatch over multiple samples.

In a probabilistic framework, predictions take the form of predictive probability distributions and therefore they need appropriate scoring rules which assign a score based on the predictive distribution and single true value (Gneiting et al. 2007). Scoring rules assess two important features of probabilistic forecasts: sharpness and calibration. Sharpness is related to the concentration of the predictive distribution and quantifies if probabilities are spread to different values or concentrated on few values. Calibration describes statistical consistency between the distributions and observed values.

Proper and strictly proper scoring rules are an important class of the scoring rules. Let $P(X)$ the predictive distribution and x the true (observed) value. We suppose that $Q(X)$ is the true distribution from which sample x was drawn. A reward of a forecaster is given by the score $S(P, x)$. Let us use $S(P, Q)$ for expected value of $S(P, \cdot)$ under Q. The *strictly proper* scoring rule is such that $S(Q, Q) \geq S(P, Q)$ with equality if and only if $P = Q$. A *proper* scoring rule satisfies $S(Q, Q) \geq S(P, Q)$ for all P and Q. Proper scoring rules encourage honest predictions: when the best estimate of P predicted by a forecaster is Q, the strategy to achieve the best score is to use the distribution Q. The forecaster has no interest in modifying Q, as it will not result in a better score.

In geostatistics, the first applications of cross-validation were mentioned by David (1976) and Delfiner (1976). Although Dubrule (1983) generalized cross-validation for kriging in the unique neighborhood case for large datasets, the cross-validation term has been used as a synonym of the leave-one-out technique. An interesting alternative cross-validation technique is the orthonormal residuals as introduced by Kitanidis (1991). In this technique, the data is ordered, and starting with one point, the consecutive points are predicted one by one and then added to the conditioning dataset. The standardized residuals (residuals divided by kriging standard errors) are computed at each step and their statistics are investigated to validate the model.

Cross-validation methods have been well established and extensively used for continuous variables and variogram-based methods (see e.g. textbooks of Chiles and Delfiner 2012; Cressie 1993), but there is not yet a similar consensus for the methods that should be used for categorical variables even if different applications of model testing techniques in the categorical case have been published (e.g. Allard et al. 2012; Madani et al. 2018).

In the framework of multiple-point statistics (MPS), the question of the training image (TI) selection and parameter identification has been treated using a

wide range of methods mainly designed to compare some characteristics of the training image with the simulations. The question of the quality check of MPS simulations is discussed in detail in Chapter 8 of Mariethoz and Caers (2015). For example, Boisvert et al. (2007) compared the distribution of runs between the training image and the simulations. Pérez et al. (2014) focused on the frequency of patterns found in the conditioning data and the training image. Tan et al. (2014) compared the patterns of the training image with those of the simulations at different scales. Rongier et al. (2016) quantified the mismatch of connectivity and geometrical metrics between the TI and the simulations. Feng et al. (2017) used a minimal distance between the data event found in the TI and in the conditioning data. Based on any of these metrics, one can derive automated parameter selection methods (Baninajar et al. 2019; Dagasan et al. 2018). However, in real case applications the problem is not necessarily to reproduce accurately the patterns found in a training image because this image is not known precisely and is itself a parameter to be identified from the conditioning data. Al-Mudhafar (2018) mentions that he uses leave-one-out and split-sample approaches for different MPS realizations of a fluvial environment. The method of Pérez et al. (2014) is the only one allowing to identify the training image. However, it assumes that the simulation is stationary, while this is rarely the case for practical applications in which there are different trends, for example in orientations, proportions of the facies, or even types of patterns. We, therefore, argue that a more realistic strategy is to apply a generic cross-validation technique to identify the training image and all the other parameters when using an MPS model.

To address the issues described above, this paper aims to present a generic methodology based on K-fold cross-validation for the categorical case. It allows ranking spatial simulation methods given some observation points. The technique is based on the mean quadratic score (also called Brier score) and is especially suitable for assessing probabilistic outcomes of a simulation method. The application of the methodology is illustrated in an MPS framework, but the approach can be used with any categorical simulation method honoring conditioning data. For demonstrating the performance of the method, we show a benchmark example of training image selection and parameter selection (including the TI as one of the parameters) in a realistic non-stationary case.

2.2 Cross-validation methodology

This section presents a cross-validation methodology for geostatistical simulations. We suppose that with the simulation method, N observations are available: they are pairs of $(\mathbf{x}_n, y_n), n = 1, \ldots, N$, where $\mathbf{x}_n$ are vectors of spatial coordinates of the n-th observation and $y_n \in \{1, \ldots M\}$ is the facies index observed at point $\mathbf{x}_n$. M is the number of possible facies. Let us denote with $\mathcal{K}$ the set of all available observations, $\mathcal{K} = \{(\mathbf{x}_n, y_n), n = 1 \ldots N\}$.

In this section, scoring rules for probabilistic forecasts will be introduced. They

allow quantifying the performance of a stochastic method when resimulating known values. Then, K-fold cross-validation methodology will be reviewed and adapted to spatial datasets.

Scoring rules

Scoring rules aim to quantify the quality of a probabilistic forecast by comparing it with a single true value. In our setting, repeating stochastic geostatistical simulations yields a probabilistic forecast of a geological facies for each point in the simulation domain. Some points in the simulation domain can be compared with the true, observed facies value. The observed facies value is a value from the set $\{1, \ldots, M\}$ and the probabilistic forecast is a probability vector $\mathbf{p} = (p_1, \ldots, p_M)$, where each vector element p_j, $j = 1, \ldots, M$ describes probabilities of different facies.

We consider scoring rules S in the form of a collection of M functions:

$$S(\cdot, i) : \mathcal{P}_M \mapsto \mathbb{R}, \qquad i = 1, \ldots, M \tag{2.1}$$

where $\mathcal{P}_M = \{\mathbf{p} = (p_1, \ldots, p_M) : p_1, \ldots, p_M \geq 0, p_1 + \ldots + p_M = 1\}$ is the probability forecasts space where p_j, $j = 1, \ldots, M$ is the forecast probability of the outcome j and i is the index of the observed value (facies).

The scoring rules apply to a single observation but in practice they are aggregated, and forecasts are ranked using average scores:

$$\bar{s}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} s, \tag{2.2}$$

where $\mathcal{S}$ is the set of scores, and $|\mathcal{S}|$ is the cardinal of $\mathcal{S}$.

It is also possible to compute the average class score, or balanced mean score. Such a balanced score generalizes average class accuracy (Kelleher et al. 2015), also called balanced accuracy (Brodersen et al. 2010), used in machine learning for assessing classifier's performance. It helps correct too optimistic estimates of classifier performance given by average accuracy and is especially useful when dealing with imbalanced datasets.

Let $\mathcal{M}$ be the set of observed facies (classes) used to compute the set of scores $\mathcal{S}$. Now let us denote $\mathcal{S}^m$, $m \in \mathcal{M}$, the subset of scores where the observed facies was m: $\mathcal{S}^m = \{s \in \mathcal{S}, \text{observed value is } m\}$. The balanced mean score gives the same importance to each facies in the dataset and is defined as the arithmetic mean of the average class scores, i.e.:

$$\hat{s}(\mathcal{S}, \mathcal{M}) = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \frac{1}{|\mathcal{S}^m|} \sum_{s \in \mathcal{S}^m} s. \tag{2.3}$$

Quadratic score

The quadratic (or Brier) score was first introduced as a measure of quality for meteorological forecasts (Brier 1950). It is a strictly proper scoring rule (Gneiting et al. 2007) given by:

$$S(\mathbf{p}, i) = -\sum_{j=1}^{M}(\delta_{ij} - p_j)^2 = 2p_i - \sum_{j=1}^{M} p_j^2 - 1, \tag{2.4}$$

with $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise. The quadratic score values are between -2 and 0, the higher the better. An ideal forecaster, the one predicting the correct outcome with the probability of 1 would score 0. The worst forecaster is the one attributing probability of 1 to a wrong class and scores -2. A better forecaster which makes fewer systematic errors, e.g. spreads probability equally to wrong classes, would have a better score than the one who prefers one wrong class to the others.

As Brier (1950) pointed out, the score encourages the forecaster to get the prediction exactly right. A sharp and correct prediction would score close to 0. On the other hand, the forecaster should state unbiased estimates of probability, when not able to forecast perfectly. The strategy to predict the most frequent facies with certainty is penalized in comparison to unbiased strategy (referred to as climatological in weather forecasting), which just learns the probabilities from the training set. In this way, the quadratic score encourages calibrated forecasts.

Zero-one score

Zero-one score is a proper but not a strictly proper scoring rule (Gneiting et al. 2007) given by:

$$S(\mathbf{p}, i) = \begin{cases} 1/\ |modes(\mathbf{p})| & \text{if } i \in modes(\mathbf{p}) \\ 0 & \text{otherwise,} \end{cases}$$

where $modes(\mathbf{p}) = \{i : p_i = \max_{j=1,\ldots,M} p_j\}$ is the set of modes of $\mathbf{p}$. Mean zero-one score values are between 0 and 1 and are easily interpreted: the mean zero-one score gives a fraction of observed points (geological facies), which were given the highest probability by the simulation method.

Linear score

A generalization of balanced accuracy for probabilistic forecasts is the balanced mean linear score with the linear score given by:

$$S(\mathbf{p}, i) = p_i.$$

This score simply attributes the probability of the true value. The advantage of the mean linear score is its intuitive interpretation: it indicates how frequently

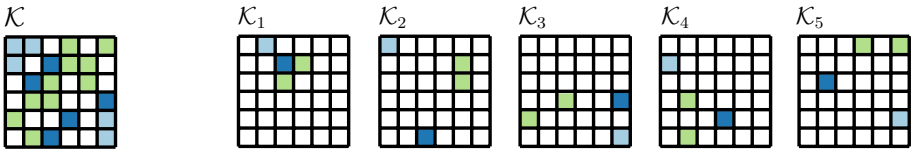

Figure 2.1: Example of shuffled 5-fold stratified split of data ($\mathcal{K}$) in a regular grid belonging to three classes (light blue, dark blue, green).

simulations predict the correct value. It corresponds to the proportion of the correct forecast and is often used in machine learning to compare several methods. However, the linear score has some undesirable properties (Selten 1998): it does not encourage fair predictions and is neither proper nor strictly proper. While the balanced linear score is still not proper, it forces the forecaster to try to represent rare facies (by increasing their share in the score) and remains easy to interpret.

Unbiased classifier

It is often useful to compare scores of geostatistical simulations with scores of a simple reference method. As already mentioned when discussing the quadratic score, in the field of weather forecasting, the climatological forecaster serves often as a reference. It takes into account averaged historical values. In a broader sense, climatological forecasts are calibrated forecasts but lack sharpness (Gneiting et al. 2007). Such a reference simulation method for spatial data would be the unbiased classifier, which looks at the proportions of facies in the complete training set and uses them to estimate the vector of probabilities (which is constant over the whole domain). The score obtained by the unbiased classifier will be referred to as the reference score.

If stratified K-fold is used (explained in the next subsection), the reference balanced mean linear score is equal to $1/M$. It is another property of the balanced mean linear score that makes it intuitive. For the mean quadratic score and mean zero-one score, the reference score will not only depend on the number of different facies M, but also on the proportions of the facies in the dataset.

K-fold cross-validation

K-fold cross-validation consists of dividing the dataset $\mathcal{K}$ into K subsets of equal sizes and performing K iterations: in each iteration, one subset is removed from $\mathcal{K}$ and becomes the validation set, while the complementary set (rest of the data) forms the training set used as conditioning data for the geostatistical method. To describe the split, we can define the partition function:

$$\kappa(\cdot) : \{1, \dots, N\} \mapsto \{1, \dots, K\}$$

that maps each point's index n to subset (iteration) index $\kappa(n) \in \{1, \dots K\}$, $n = 1, \dots, N$. We define K disjoint sets $\mathcal{K}_1, \dots, \mathcal{K}_K$:

$$\mathcal{K}_k = \{(\mathbf{x}_n, y_n) : \kappa(n) = k, \quad n = 1, \dots, N\}, \quad k = 1 \dots K.$$

The union of these sets is the dataset $\mathcal{K}$. The partition should be made in such a way that $|\mathcal{K}_1| = \dots = |\mathcal{K}_K|$. Since geological datasets are often imbalanced (e.g. proportions of facies are strongly different, rare facies are present), it is important to use stratified cross-validation. In stratified cross-validation, the dataset is split into subsets that have the same proportion of classes (facies). This translates to the following condition: $|\mathcal{K}_1^m| = \dots = |\mathcal{K}_K^m|$, $m = 1, \dots, M$, where $\mathcal{K}_k^m$ denotes the subset of $\mathcal{K}_k$ containing only samples of category m:

$$\mathcal{K}_k^m = \{(\mathbf{x}_n, y_n) : \kappa(n) = k, \quad y_n = m, \quad n = 1, \dots, N\}, \quad k = 1, \dots, K.$$

Moreover, if observation points are not spatially correlated, the split should be made in a random way, for instance by shuffling (randomly reordering) the data first, as depicts Figure 2.1.

The training dataset is the complementary dataset:

$$\bar{\mathcal{K}}_k = \{(\mathbf{x}_n, y_n) : \kappa(n) \neq k, \quad n = 1, \dots, N\}, \quad k = 1, \dots, K,$$

and it becomes the conditioning data for the geostatistical method (Figure 2.2). For each iteration $k = 1, \dots, K$, the geostatistical method produces probability vectors for each point in the validation set $\mathcal{K}_k$ and uses $\bar{\mathcal{K}}_k$ as conditioning data. Let $\hat{f}$ be the geostatistical estimator $\hat{f}(\cdot, \bar{\mathcal{K}}_k) : \mathbb{R}^3 \to \mathcal{P}_M$ mapping a coordinate vector to a probability vector, given $\bar{\mathcal{K}}_k$ as the conditioning set. We define the prediction $\mathbf{p}_n$ at the n-th point:

$$\mathbf{p}_n = \hat{f}(\mathbf{x}_n, \bar{\mathcal{K}}_{\kappa(n)}), n = 1, \dots, N. \tag{2.5}$$

We note here that in the case of stochastic simulation methods, it will be necessary to repeat the simulations at each iteration with the same conditioning data to obtain probabilities of categorical variables (geological facies) at each validation point (Figure 2.3).

Then a scoring function S as defined in Equation 2.1 is applied to all points in the validation set and the mean score is computed either using mean (Equation 2.2) or balanced mean (Equation 2.3). The procedure is repeated for each of the K iterations so that each subset becomes a validation set once. The mean cross-validation score CV is the average of the mean scores over all iterations. If standard mean is used:

$$CV = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|\mathcal{K}_k|} \sum_{(\mathbf{x}, y) \in \mathcal{K}_k} S(\hat{f}(\mathbf{x}, \bar{\mathcal{K}}_k), y),$$

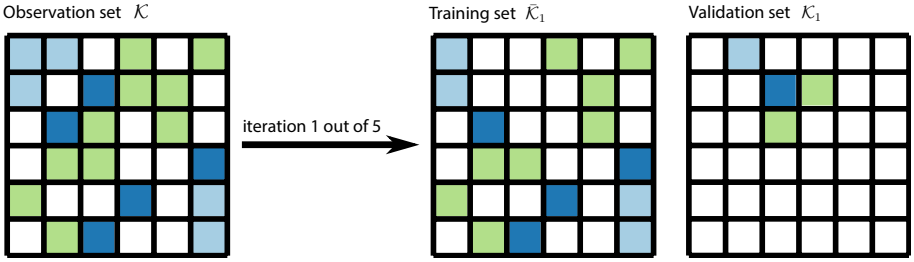

Figure 2.2: The split of the example data ($\mathcal{K}$) in a regular grid in the first cross-validation iteration. The first subset ($\mathcal{K}_1$) becomes the validation set, and the complementary dataset (training set, ($\bar{\mathcal{K}}_1$) becomes the conditioning data for the geostatistical model.

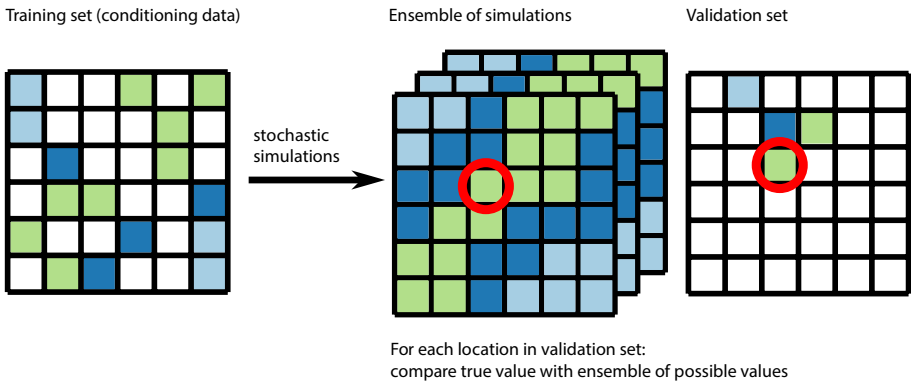

Figure 2.3: Example of one cross-validation iteration. The domain is simulated n_r times with conditioning data (ensemble of simulations is obtained). Each point in the validation set is compared with vector p constructed by aggregating realizations in the ensemble.

and in the case of balanced mean it becomes (with M_k number of different facies in $\bar{\mathcal{K}}_k$):

$$CV = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{M_k}\sum_{\substack{m=1\\ |\mathcal{K}_k^m|\neq 0}}^{M_k}\frac{1}{|\mathcal{K}_k^m|}\sum_{(\mathbf{x},y)\in\mathcal{K}_k^m} S(\hat{f}(\mathbf{x},\bar{\mathcal{K}}_k),y).$$

We can now summarize the K-fold cross-validation framework in the form of an algorithm. It takes as input: K — number of cross-validation iterations; $\mathcal{K}$ — dataset of pairs $(\mathbf{x},y)$; S — scoring rule; *balance* — TRUE or FALSE indicating whether the balanced mean should be used; n_r — number of stochastic simulation runs needed to construct the probability vector $\mathbf{p}$ (Equation 2.5).

CROSS-VALIDATE($K, \mathcal{K}, S$, *balance*, n_r)

```
Shuffle randomly the data in 𝒦
Divide 𝒦 into K subsets 𝒦_1, ..., 𝒦_K
for k = 1 to K
    Use 𝒦̄_k (training set) as conditioning data
    Run geostatistical simulation n_r times
    𝒮 = ∅, ℳ = ∅
    for (x, y) ∈ 𝒦_k // for each point in validation set
        // Construct probability vector p and compute the score
        p = f̂(x, 𝒦̄_k)
        𝒮 = 𝒮 ∪ {S(p, y)}
        ℳ = ℳ ∪ {y}
    if balance is TRUE
        s_k = ŝ(𝒮, ℳ)
    else
        s_k = s̄(𝒮)
return (1/K) Σ_{k=1}^{K} s_k
```

We used $\hat{s}$ to refer to mean balanced score (Equation 2.3), and $\bar{s}$ to mean score (Equation 2.2). The cross-validation requires n_r geostatistical model runs per iteration, thus Kn_r model runs in total.

2.3 Case studies

The proposed cross-validation methodology can be used with any stochastic simulation method but it will be tested on two MPS cases. Therefore we also introduce the basic notions of MPS and the Direct Sampling algorithm. The two examples are: a training image selection problem and a parameter selection problem. Both are 2D categorical synthetic set-ups.

Direct Sampling

MPS algorithms use a conceptual geological model, provided as a training image (TI). The TI is an implicit database of patterns and an example of how the simulated field should look like. Direct Sampling (Mariethoz et al. 2010c) is a point-based method and constructs fields by filling the regular simulation grid point by point. It honors the conditioning data by simply putting them in the simulation grid, It can perform multivariate simulations and supports transformations, such as field rotations or affinities (Mariethoz and Kelly 2011). In this work, DeeSse implementation of Direct Sampling is used (Straubhaar et al. 2020).

Direct Sampling is controlled by three main parameters: number of nearest neighbors (n), distance threshold (t) and maximal fraction scan (f). These parameters are crucial for the quality of simulated fields and the computing time.

Their tuning may be challenging, since the results depend on the complexity of the TI as well as the interaction between the parameters and the patterns in a complicated manner (Meerschman et al. 2013). The number of nearest neighbors limits how many pixels are included in a pattern during the pattern search. Typical values range from several to one hundred or more. As a rule of thumb, the more neighbors, the better the quality of simulations, but also the higher the computational cost. The distance threshold specifies the maximum acceptable dissimilarity between the conditioning pattern and patterns found in the TI, defined as the proportion of mismatching nodes in the two patterns. It can range from 0 to 1. The value 0 means that only a perfect match is accepted, while with the value of 1, every pattern is suitable. This parameter typically ranges from 0.01 to 0.1. The maximal scan fraction indicates what fraction of the training image can be potentially scanned before the search is stopped (which can happen if no sufficiently good match was found: in such a case the best candidate found so far is accepted). The value of 1 means that the whole TI can be scanned and a value close to 0 would mean that the scan is stopped after scanning only one node. The scan fraction helps avoid the verbatim copy of the TI and limit the computation time.

Benchmark set-up for training image selection

The first example is a benchmark training image selection problem, which was first published by Pérez et al. (2014) and also used in the work of Feng et al. (2017). In this set-up, there are three training images with different features: ellipsoids, sine waves, and vertical stripes (Figure 2.4). Pérez et al. (2014) used the same training image generator tool to construct both the reference training images and synthetic realities. In our set-up a different approach for constructing synthetic realities was used. To allow more pattern variability, we first performed one unconditional DeeSse simulation with each of the TI and the following parameters: $n = 60$, $t = 0.05$, and $f = 0.25$ (Figure 2.5). Then, we sampled points from the synthetic realities with 10 different sampling rates ranging from 0.0025 to 0.16. In this way, we created 30 synthetic observation sets: 3 TI and 10 numbers of points ranging from 25 to 1600. Figure 2.6 shows examples of such observation sets containing 1600 points each. The sets corresponding to the different number of samples are sampled independently.

Roussillon plain synthetic example

In the second example, an alluvial aquifer located in the Roussillon plain near Perpignan (France) is considered. It is a simplified 2D version of the model build by Dall'Alba et al. (2020). The area is modeled considering four geological facies: river bed, crevasse splays, flood plains, and alluvial fans. The position of the facies in the basin is guided using a trend defined over the area (Figure 2.7a) as well as in the TI (Figure 2.8). The geological features are oriented according to the paleotopography of the region (Figure 2.7b). In order to test the methodology with a known reference, a synthetic reality was constructed by

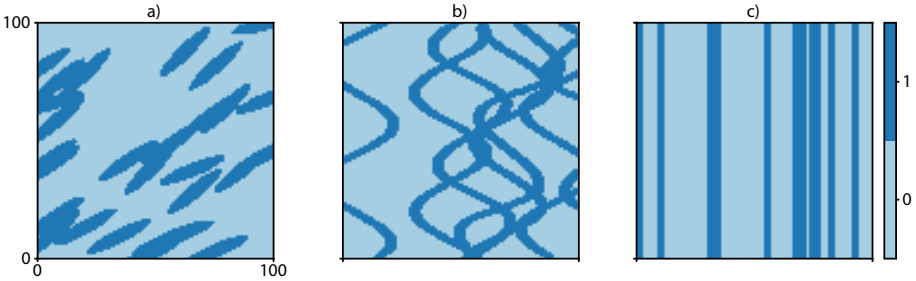

Figure 2.4: Three training images of type a (ellipsoids), b (waves), c (stripes) used in the benchmark of training image selection (data from Pérez et al. (2014)).

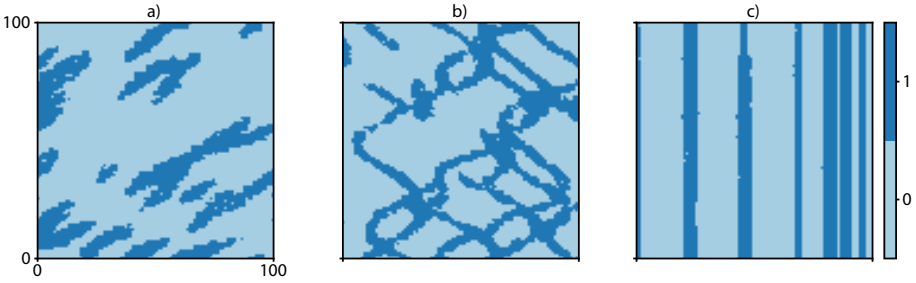

Figure 2.5: Three synthetic realities, each obtained by means of DeeSse simulation using training images a, b, c, respectively.

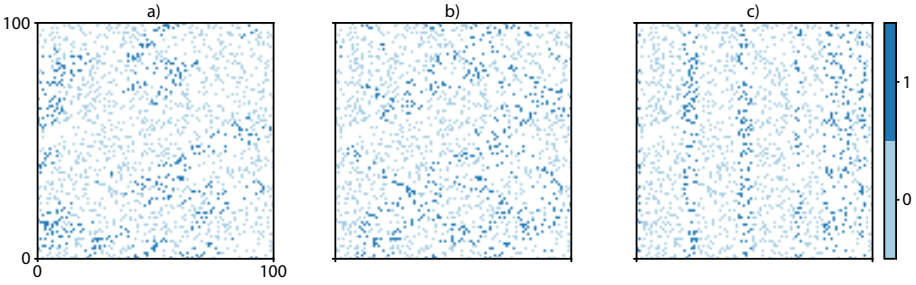

Figure 2.6: Example of synthetic observation sets obtained by sampling from each of the synthetic realities a, b, c. Here each synthetic observation set contains 1600 samples.

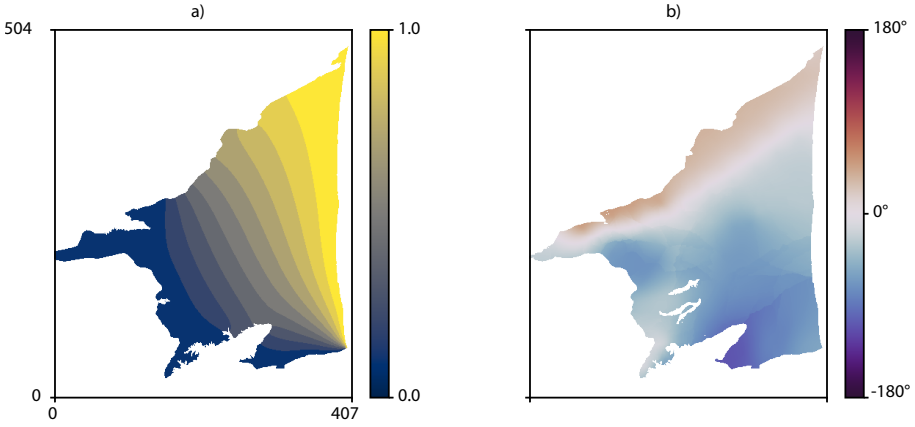

Figure 2.7: Area of the multivariate simulation with defined trend (a) and orientation (b).

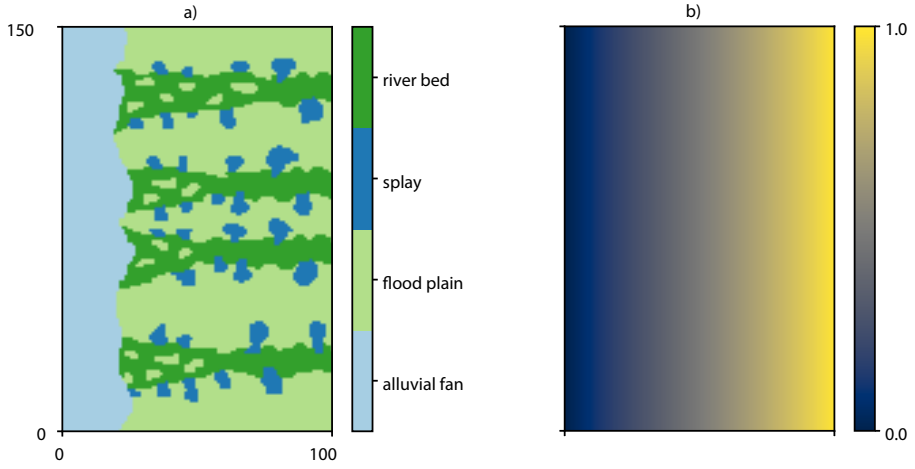

Figure 2.8: The reference training image (a) with the defined trend (b). The training image contains 4 geological facies.

performing an unconditional DeeSse simulation with the following parameters: $n = 50$, $f = 0.5$, $t = 0.05$. Then 50, 150, and 600 random samples were drawn from the area to form 3 synthetic observation sets (Figure 2.9). These numbers correspond to the number of pumping wells (50), wells with lithology information (150), and all wells (600) in the area. The example consists in selecting the best DeeSse parameters (including the TI) for each of the observations sets. Two candidate training images will be considered: the training image used to construct the observation (reference TI, "true TI") set and the analog training image (Figure 2.10).

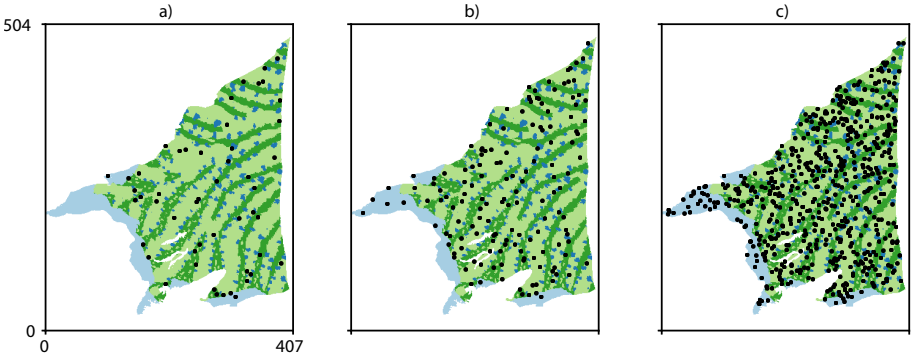

Figure 2.9: The synthetic reality obtained by DeeSse simulation using the reference TI and samples locations: (a) 50 wells, (b) 150 wells, (c) 600 wells, representing synthetic observation datasets.

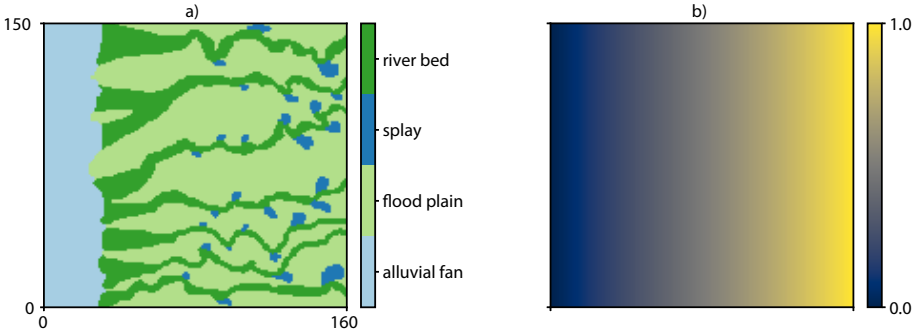

Figure 2.10: The analog training image (a) with its corresponding trend (b). The training image contains 4 geological facies.

2.4 Results

The higher the cross-validation score, the better the predictive power of the method. In our setting, the higher cross-validation score points to better simulation parameters or a more compatible training image. The results in this section were obtained using the 5-fold cross-validation. Three scores were compared for each run: quadratic (Brier) score, zero-one score, and balanced linear score. Since at each cross-validation iteration, simulations are repeated to construct the probability vector $\mathbf{p}$, the technique has one hyper-parameter, which was adjusted in the first test case: the number of realizations per experiment. It defines how many times a simulation is repeated to approximate the probability distribution of facies. The minimal number is 1 and it would correspond to a deterministic approach.

Training image selection

For each observation set, we performed the cross-validation using the original DeeSse parameters ($n = 60$, $t = 0.05$, and $f = 0.25$) and compared the cross-validation scores for each of the TI.

Firstly, to check the influence of the number of realizations per experiment, we fixed the sampling rate to 0.005, thus used the datasets of the three types with 50 samples each. For each of the datasets, we varied the number of realizations per iteration (ranging from 1 to 49) and recorded cross-validation scores using each of the training images a, b, and c and different scores: the mean quadratic score, the mean zero-one score, and the mean balanced linear score (Figure 2.11). The figure also reports reference scores. Even for a small number of realizations, the highest cross-validation scores are attributed to the correct training image, except for the case b. For the quadratic scoring rule, a small number of realizations results in generally lower scores. This is expected since a small number of realizations implies that the probability densities $\mathbf{p}$ are estimated from a small sample, and can result in a low score if predictions are incorrect. Adding more realization gives more samples to estimate more accurately the spread of the probability distribution and the uncertainty resulting in a better quadratic score. In the case of the balanced linear score there is no such effect, in the case b we see an inverse trend: score values slightly decrease with the number of realizations. This is probably related to the nature of the score, not sufficiently penalizing wrong predictions with high probabilities. The mean zero-one score seems to be the most sensitive to the number of realizations among the three scoring methods. It provides scores that are less robust than the other methods and fluctuates erratically when the number of realizations is modified. This is not surprising because it accounts only for the maximum probability value and not for the complete pdf. Scores obtained using quadratic rule suggest that 30 realizations per iteration are sufficient to obtain robust results. Therefore, we fixed this parameter to this value for the remaining tests.

The observation sets a and c are characterized by a large difference between the best training image and the other training images. The most compatible TI has also a significantly higher score than the reference score. However, it does not apply to the case b. The mean quadratic score using the corresponding training image is around the reference value. Similar behavior is seen in the case of the zero-one score. The mean balanced score gives more optimistic results, higher values than the reference. While the mean quadratic score and the mean balanced linear score have correctly identified the most compatible training image in the case b, the low values (compared to the reference score) and the small difference between the different TIs suggest that the dataset is too small for a reliable choice of the best training image.

Secondly, with the fixed number of realizations, cross-validation was run for

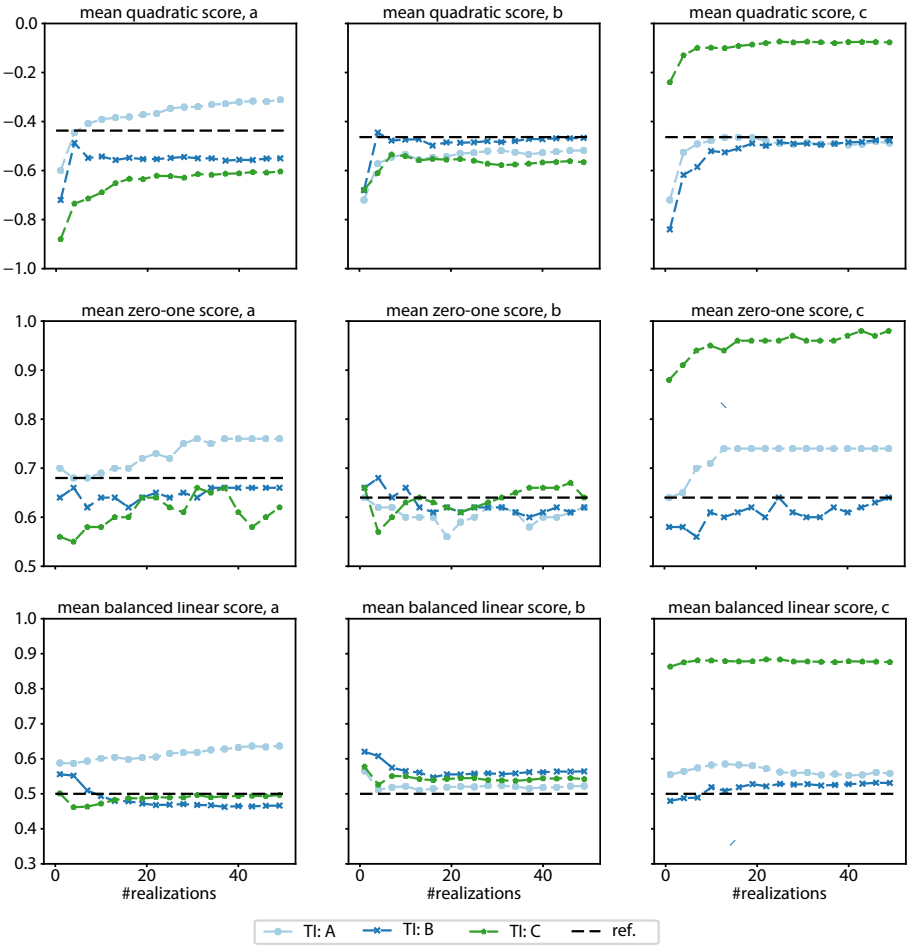

Figure 2.11: Sensitivity study with respect to the number of realizations per fold for each of the different types of observation sets a, b, c. Each of the samples contained 50 observations. Mean quadratic score, mean zero-one score and mean balanced linear score were used.

Parameter	values
f	0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.4, 0.8
n	40, 20, 10, 5
t	0.01, 0.05, 0.1, 0.2

Table 2.1: DeeSse parameters, which were tested in the parameter selection for the multivariate simulation example. For each combination of these parameters with both training images a cross-validation score was computed.

each observation set. Figure 2.12 shows the mean quadratic scores, the mean zero-one scores and the mean balanced linear scores for different numbers of samples in observation sets. The higher score corresponds to better TI compatibility. In all cases, the original ("true") training image was correctly identified by the mean quadratic score and the mean balanced linear score except for one dataset: that of type b with 25 samples. The mean zero-one score was not able to correctly identify the most compatible TI in the case of type b with 25 and 50 samples. These results suggest that this sparse dataset is not sufficient to identify the synthetic reality and the mean quadratic scores close to the reference seem to confirm this statement. The mean zero-one and linear balanced scores attributed much higher values to the training image c for this observation set.

For larger observation sets, all scores tend to improve irrespectively of the TI. It might seem surprising at first, but can be explained by the fact that all structures have some short-range continuity. Direct Sampling (and all interpolation or simulation methods in general) respects this short-range continuity. This results in better predictions for points in the vicinity of observed data and there are more such points in larger observation sets and therefore the predictions are better.

Parameter selection

To make the example closer to a real scenario, we consider the simulation parameters as unknown. The cross-validation scores were therefore computed with two candidate training images: the reference one (Figure 2.8) and the analog one (Figure 2.10) and for all combinations of the Direct Sampling parameters in Table 2.1. The number of realizations per iteration was fixed to 30. We note that the parameter set used to generate the synthetic reality is not present in the proposed set of combinations.

For each synthetic observation set (with 50, 150, and 600 points), cross-validation scores were obtained using three scoring methods: mean quadratic score, mean zero-one score, and mean balanced linear score. Table 2.2 presents the best scores for each observation set, for each TI and for each scoring method. Corresponding DeeSse parameters are also reported along with ref-

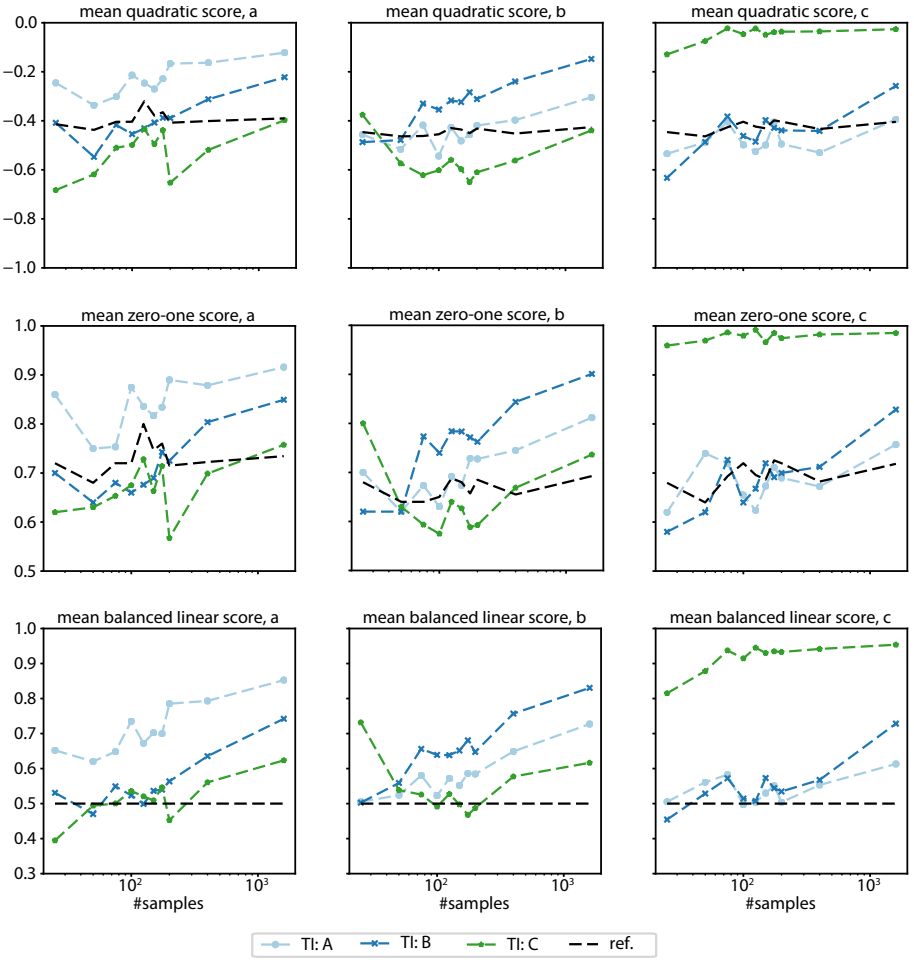

Figure 2.12: Cross-validation scores for synthetic observations with respect to the number of samples in the dataset for each of the different types of observation sets a, b, c. 30 realizations per fold were generated. Mean quadratic score, mean zero-one score, and mean balanced linear score were used.

wells	function	TI	score	reference	t	f	n
50	brier	ref.	-0.47	-0.50	0.05	0.200	40
50	brier	analog	-0.46	-0.50	0.05	0.200	40
50	zero-one	ref.	0.72	0.68	0.05	0.400	40
50	zero-one	analog	0.72	0.68	0.05	0.020	40
50	linear	ref.	0.48	0.27	0.05	0.020	40
50	linear	analog	0.48	0.27	0.05	0.005	40
150	brier	ref.	-0.40	-0.51	0.05	0.400	40
150	brier	analog	-0.44	-0.51	0.01	0.400	20
150	zero-one	ref.	0.73	0.67	0.05	0.800	40
150	zero-one	analog	0.71	0.67	0.01	0.050	10
150	linear	ref.	0.53	0.25	0.10	0.200	40
150	linear	analog	0.48	0.25	0.10	0.020	5
600	brier	ref.	-0.28	-0.58	0.01	0.400	20
600	brier	analog	-0.38	-0.58	0.01	0.050	10
600	zero-one	ref.	0.82	0.60	0.01	0.400	20
600	zero-one	analog	0.74	0.60	0.01	0.050	10
600	linear	ref.	0.63	0.25	0.05	0.400	20
600	linear	analog	0.55	0.25	0.01	0.050	10

Table 2.2: Best cross-validation scores and corresponding DeeSse parameters for each observation set (with different number of wells), each candidate TI, and each scoring method. Corresponding best scores for optimal parameters are given.

erence scores. The example DeeSse simulations with the corresponding best parameters are shown in Figure 2.13 (reference TI) and Figure 2.14 (analog TI).

In the case of 50 wells, the reference and the analog TIs received similar scores, only slightly better than the reference score (except for the optimistic balanced linear score). It suggests that this dataset is too sparse to let us reliably choose the best DeeSse parameters. In the case of 150 wells, all the scores attributed the higher value to the reference training image. The mean quadratic score also is higher than in the case with 50 wells. In the case of 600 wells, all scores point to the reference training image and are significantly better than the reference score.

When looking at the example simulations with the reference training image with 50 and 150 wells (Figure 2.13), we observe that the parameters found by using the quadratic score and the zero-one rules result in realizations that better represent crevasse splays than those found by using the balanced linear scores.

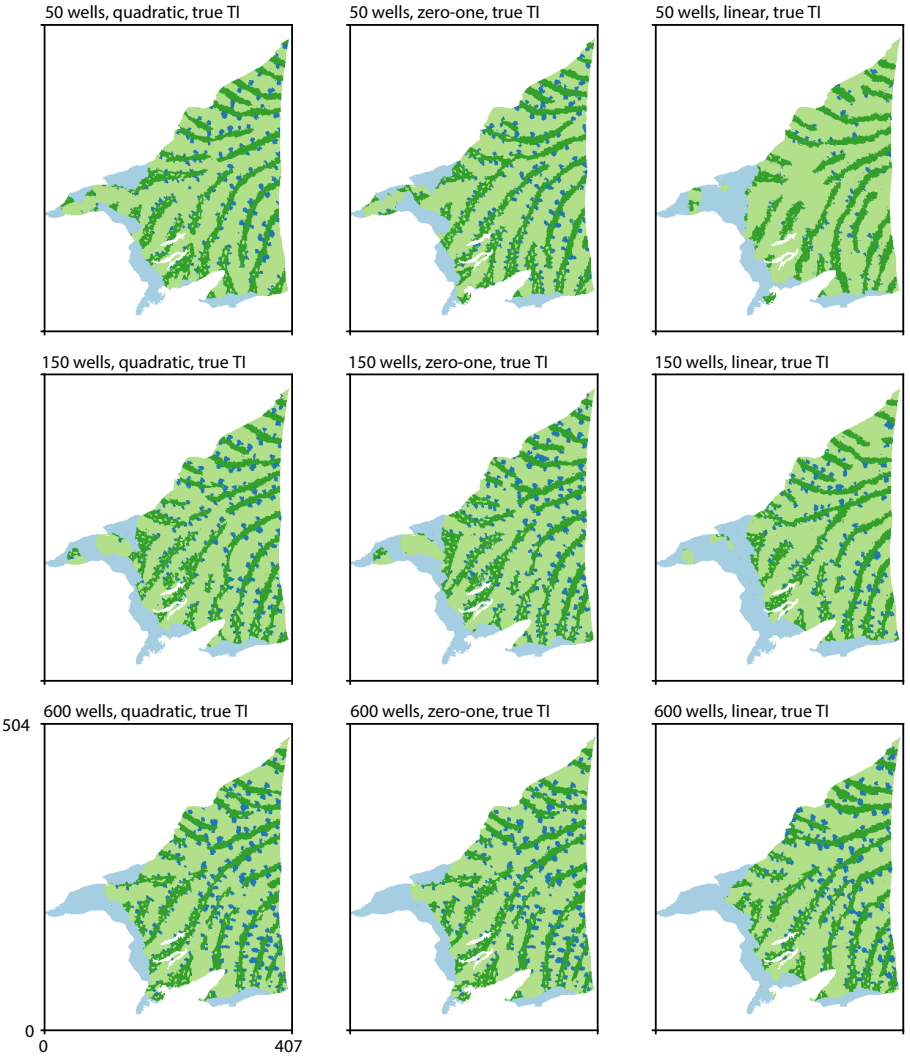

Figure 2.13: Example simulations with the reference TI, the best DeeSse parameters found with cross-validation for each observation set (50, 150, 600 wells), and different scoring methods (quadratic score, zero-one score, balanced linear score).

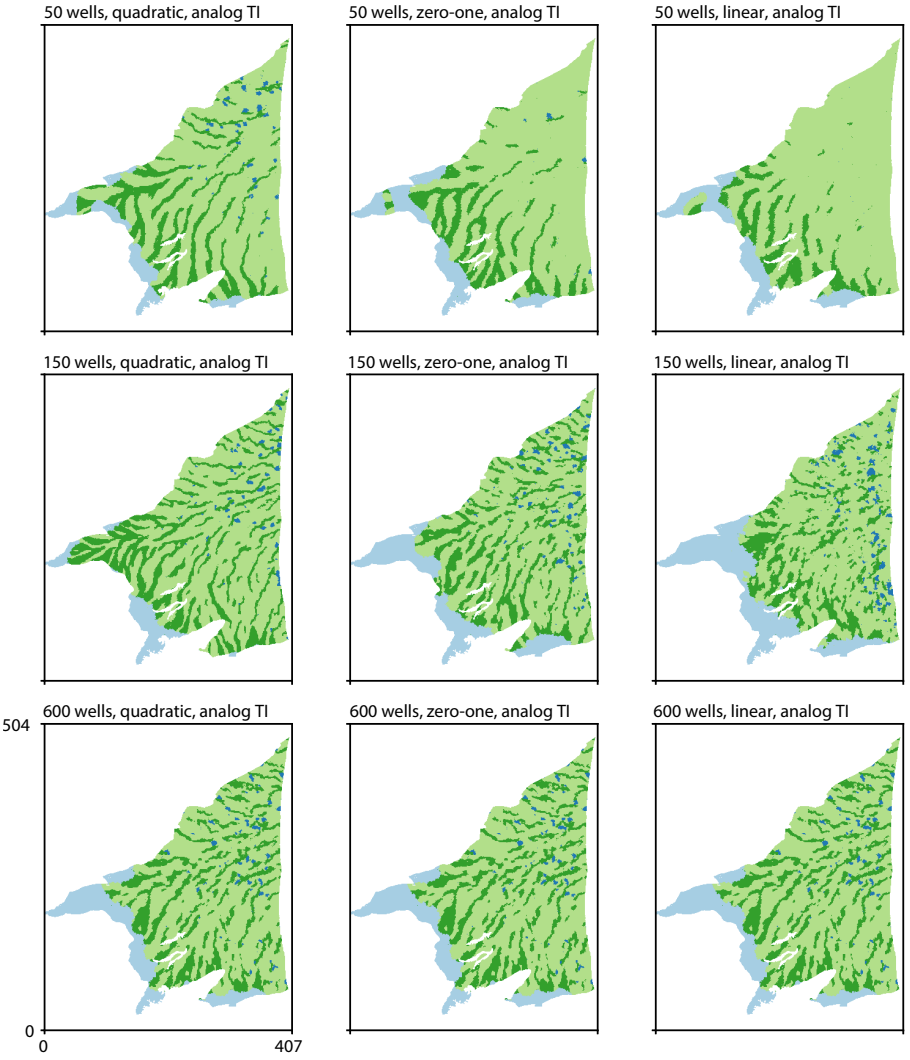

Figure 2.14: Example simulations with the analog TI, the best DeeSse parameters found with cross-validation for each observation set (50, 150, 600 wells), and different scoring methods (quadratic score, zero-one score, balanced linear score).

In the case of simulations with the analog training image (Figure 2.14), the resulting simulations are characterized by a poor reproduction of the structures of the TI. The parameters obtained from the cross-validation (a small scan fraction and a small number of neighbors) permit a high variability between the simulations but sacrifice the quality of the TI reproduction. Indeed, honoring the conditioning data requires departing from the TI which is not the one used to generate the synthetic reference.

These observations are confirmed by the analysis of Table 2.2. It shows that the best parameters for the simulation with the analog (wrong) training image and with a large conditioning dataset correspond to values that typically provide simulations that do not reproduce precisely the patterns of the training image (for example the small number of neighbors). It results in more variability and uncertainty in the predictions which is favored by the quadratic score.

2.5 Discussion

The first test case shows that the methodology is able to identify correctly a training image if the amount of conditioning data is sufficient. Moreover, using the second test case, we demonstrated that the proposed methodology accounts properly for non-stationarity because it is only based on the conditioning data and the simulated values. The main novelty of our approach is that it does not compare the patterns of the training image directly with the observed data. Often the uncertainty in the training image can be important and the best prediction can then require departing from the TI. This was already shown by Dagasan et al. (2018) but in a specific configuration, where observations were spaced regularly on a cartesian grid. The method proposed in this paper is more general. Not only it tunes the parameters with respect to the data and not the training image (as in Baninajar et al. 2019), but also treats the pattern reproduction implicitly. Respecting the patterns in the data and allowing sufficient variability is essential to produce correct predictions. Recent developments (Abdollahifard et al. 2019) aim to quantify variability and pattern consistency but with respect to the TI, not to the data. The advantage of our approach is that we no longer need to compare realizations with the training image and define quality indicators as in the works of Meerschman et al. (2013) or Rongier et al. (2016). We can see that in the second test case when using the reference training image, the simulations obtained with optimal parameters are visually comparable to the synthetic reality, and patterns of the training image (like channels continuity) are well represented.

The computation time is easy to predict, the framework requires $n_r K$ model runs. For complex geological models, this cost can be significant. In such cases, cross-validation iterations should be parallelized, as they are independent. It is also possible to parallelize the computation of realizations per iteration. To further reduce computing time, a smaller number of realizations per iteration, n_r, can be chosen. It might be possible to rank models with n_r in the range from 1

to 5 but for reliable results, at least 10 realizations per iteration should be obtained. In this work, we used 30 realizations per fold, as models were relatively cheap to compute. It is widely accepted that 5-fold or 10-fold approach are the best choice (Kohavi 1995; Rodriguez et al. 2010). It is possible that for sparse geological datasets other K are preferable but we found in this paper that the 5-fold approach performs reasonably well. Another way to reduce computing time is to simulate only points in the validation set (e.g. avoid generating the entire field). In such a case, additional tests should be made to assess the robustness of the method, since most of the geostatistical methods depend on the simulation path. Moreover, if a method can directly estimate the probability vector, without repeating the simulation, it will reduce computational time.

The same cross-validation strategy could be applied to continuous variables; in that case, the scoring function needs to be adapted. The continuous ranked probability score (CRPS) is a proper scoring rule and a counterpart of the Brier score for continuous variables (Gneiting et al. 2007). In the second test case, we used the grid search to evaluate all parameter combinations to find the optimal ones. Since the cross-validation score is a single value, it can be used as an objective function in any optimization algorithm, which could more efficiently explore the parameter space to find the best parameters of a geostatistical method.

2.6 Conclusion

In this paper, we propose a cross-validation framework that can be applied for ranking geostatistical stochastic simulation methods of categorical variables when an observation dataset is available. The method can be used for various purposes, such as selecting the best parameter set, or the best training image, even when the simulation is not stationary. It can also be employed to compare the performances of different geostatistical algorithms.

We used a stratified 5-fold approach with shuffling, our observation points were assumed to be not strongly correlated. In the case of strong correlations (e.g. a dataset with many consecutive points along wells), it should be considered to group the points (per well, for example) and then split the whole groups into cross-validation iterations.

The mean quadratic score should be used as the most reliable indicator of method performance. The mean zero-one score and balanced linear score are more intuitive but only the mean quadratic score can correctly assess the sharpness and calibration of the model. It can also be compared to the reference score obtained by using the marginal proportions of the facies as a predictor for all locations without accounting for the spatial correlation. A score lower than this reference indicates that the probabilities estimated by the model are biased.

The K-fold cross-validation framework is parsimonious in parameters. The number of realizations per iteration, n_r, controls the precision of results and the computation time; a value of n_r between 10 and 30 is suggested for generating a robust score.

The methodology can be applied to any stochastic simulation tool. While our examples used a regular cartesian grid, the cross-validation method can be applied to any grid: only a set of spatial observations is required, and an interface with a conditional simulation tool which returns the simulated value at given coordinates.

Finally, all input data, the results, and the code used to generate them are available in the repository (`https://doi.org/10.5281/zenodo.3901494`).

Chapter 3

Direct Sampling Best Candidate

3.1 Introduction

Groundwater flow and transport depend on subsurface parameters, and representation of their heterogeneity is crucial for hydrogeological applications. For example, the connectivity of geological facies (rock types) influences contaminant transport. Two-point geostatistics often cannot reproduce complex heterogeneity patterns, therefore multiple-point statistical methods have been developed to model heterogeneous discrete or continuous fields. These methods are able to represent complex patterns and use analog data (in the form of an image) as an example of the field that should be simulated. The image, called the training image (TI), acts an implicit database for multiple-point patterns.

There exist many MPS simulation tools (Mariethoz and Caers 2015), but one of the most prominent MPS algorithms is Direct Sampling (DS), since it offers a great flexibility and is computationally efficient (Mariethoz et al. 2010b). DS is suitable for many use cases due to its rich features such as the simulation of non-stationary fields, accounting for continuous maps of rotations or affinity ratios (Mariethoz and Kelly 2011), trends expressed as secondary variables to guide the patterns in the simulation grid, multi-resolution constraints on the patterns (Straubhaar et al. 2020), or handling complex conditioning data such as inequality constraints (Straubhaar and Renard 2021). The parallelization of DS (Mariethoz 2010) is another advantage, allowing to run efficiently for large simulation grids. DS has been used and tested in some hydrogeological inversion frameworks, such as posterior population expansion (Jäggli et al. 2017; Jäggli et al. 2018), iterative spatial resampling (Mariethoz et al. 2010c), and Ensemble Smoother with multiple data assimilation (Lam et al. 2020). The applications of DS go beyond hydrogeology, they also include reservoir modeling (petroleum engineering), ore reserve estimation (Dagasan et al. 2018), climate modeling, glacier bedrock simulations (Neven et al. 2021).

As with any geostatistical algorithm, DS has computational parameters which

govern simulation quality and computation time. The choice of optimal simulation parameters can be computationally expensive, especially when a quality-performance equilibrium is sought. While some rules of thumb have been proposed for choosing DS parameters (Meerschman et al. 2013), a systematic search for optimal parameters can have a significant computational cost (Dagasan et al. 2018). This is a consequence of the fact that there are three computational parameters that require tuning: the distance threshold (t), the maximal scan fraction (f), and the number of nearest neighbors (n). The distance threshold controls how close patterns must be when searching for patterns, the maximal scan fraction limits the search in the TI, and the number of nearest neighbors controls the size of the patterns. The quality of the simulations and the simulation time depend on the choice of these parameters in a complex manner, therefore an extensive search in parameter space is often the only choice, when one wants to avoid the trial-and-error approach. Another difficulty lies in the fact that these parameters are often not very intuitive for new DS users.

To address these points, we propose a modified Direct Sampling algorithm, Direct Sampling Best Candidate (DSBC), which is equivalent to running DS with the distance threshold set to 0. It can be used in two ways: either as an implementation strategy and a different algorithm which can also have features of DS; or it can be understood as a parameter tuning strategy and used with existing DS implementations. In this paper, we focus on DSBC as a parameter tuning strategy and present test cases that show the advantages of such a simplification. Only two computational parameters must be tuned; their influence on the computation time is predictable. The extensive search for optimal parameters now requires only two dimension, which means that parameters can be chosen with more granularity and more rapidly. Three test cases validate the DSBC approach: a categorical unconditional simulation, a continuous simulation with conditioning data and a categorical simulation with trend and orientations.

The chapter is structured as follows. First, we present the Direct Sampling and Direct Sampling Best Candidate (DSBC) methods and compare their algorithmic properties. Second, quality metrics are presented which will be used to compare the performance of the two algorithms. Third, the test cases are presented. Fourth, results and comparison of the DS vs DSBC are shown. Finally, we wrap up with conclusions.

3.2 Direct Sampling Best Candidate

We will review here the Direct Sampling (DS) algorithm, which is the base of Direct Sampling Best Candidate (DSBC). We will follow a slightly modified notation as in the original DS paper (Mariethoz et al. 2010b).

The aim of DS is to simulate a random field $Z(\mathbf{x})$ on a simulation grid (SG).

The different locations (pixels, nodes) on the grid will be denoted with $\mathbf{x}$. All the non-informed locations $\mathbf{x}$ in the simulation grid will be iteratively visited and filled by the algorithm. The simulation grid can also be partially filled prior to running the algorithm. In such a case, we call the informed nodes the conditioning data. DS uses a training image (TI) as an analog (example) random field $Z(\mathbf{y})$, which is a model for the field $Z(\mathbf{x})$. The algorithm takes the training image as one of the input parameters. We will use $\mathbf{y}$ to indicate the nodes in the training image. The conditioning data is a set of pairs $\{(\mathbf{x}_1^{HD}, z_1), (\mathbf{x}_2^{HD}, z_2), \dots (\mathbf{x}_{N_{HD}}^{HD}, z_{N_{HD}}), \}$, where $\mathbf{x}_i^{HD}$ denotes the position of ith hard conditioning data, z_i its corresponding value, and N_{HD} is the number of conditioning data points. For example, it might be an information that at some location, a certain geological facies is found (e.g. from borehole data). Conditioning data is optional.

After migrating conditioning data to the simulation grid, the remaining points which have not been filled, are ordered $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, where $N = N_{SG} - N_{HD}$, and N_{SG} number of nodes in the simulation grid. This ordering is called a simulation path, and it is usually random. If no conditioning was used, the value of the first point is taken randomly from the TI: $Z(\mathbf{x}_1) = Z(\mathbf{y})$, where $\mathbf{y}$ is a random point in the TI.

DS fills the simulation grid node by node. Let i be the index of the next node to be filled. First, n closest neighbors of $\mathbf{x}_i$ are found. The neighbors are the nodes which have already been filled. In the early iterations (i small), it might happen that fewer than n neighbors can be found. Then, all of them are considered, but for simplicity in notation let us suppose that n neighbors were found and denote them with:

$$\mathcal{X}_i = \{\mathbf{x}_i^1, \mathbf{x}_i^2, \dots, \mathbf{x}_i^n\}. \tag{3.1}$$

Second, the set $\mathcal{L}_i$ of lag vectors is found:

$$\mathcal{L}_i = \{\mathbf{h}_j : \mathbf{h}_j = \mathbf{x}_i^j - \mathbf{x}_i, \quad j = 1, \dots, n\}. \tag{3.2}$$

Before we proceed to the third step, we need definitions of the neighborhood, data event, the search window, and the distance function. The neighborhood $\mathcal{N}$ of the node $\mathbf{x}$ given the lag vectors $\mathcal{L}$ is defined by:

$$\mathcal{N}(\mathbf{x}; \mathcal{L}) = \{\mathbf{x} + \mathbf{h}_1, \mathbf{x} + \mathbf{h}_2, \dots \mathbf{x} + \mathbf{h}_n\}, \tag{3.3}$$

and data event at the node $\mathbf{x}$ given the neighborhood $\mathcal{L}$:

$$\mathcal{D}(\mathbf{x}; \mathcal{L}) = \{Z(\mathbf{x} + \mathbf{h}_1), Z(\mathbf{x} + \mathbf{h}_2), \dots, Z(\mathbf{x} + \mathbf{h}_n)\}. \tag{3.4}$$

The search window $\mathcal{Y}$ is the set of points in the TI, which can be visited to look for the matching pattern:

$$\mathcal{Y}(\mathcal{L}) = \{\mathbf{y}_j \in TI : \text{for all } \mathbf{y} \in \mathcal{N}(\mathbf{y}_j, \mathcal{L}) \text{ such that } \mathbf{y} \in TI\}. \tag{3.5}$$

The distance between two data events can be defined in various manners. Different metrics are used for categorical and continuous variables. The distance only make sense, when the same lag vectors are used. The default categorical distance reads:

$$d(\mathcal{D}(\mathbf{x}), \mathcal{D}(\mathbf{y}); \mathcal{L}) = d(\mathcal{D}(\mathbf{x}; \mathcal{L}), \mathcal{D}(\mathbf{y}; \mathcal{L})) = \frac{1}{n} \sum_{i=1}^{n} \left(1 - \mathbb{1}_{Z(\mathbf{x}+\mathbf{h}_i)}(Z(\mathbf{y}+\mathbf{h}_i))\right) \tag{3.6}$$

with the indicator variable: $\mathbb{1}_x(y) = 1$ if $x = y$ and 0 otherwise. The default continuous distance reads:

$$d(\mathcal{D}(\mathbf{x}), \mathcal{D}(\mathbf{y}); \mathcal{L}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \frac{\left[Z(\mathbf{x}+\mathbf{h}_i) - (Z(\mathbf{y}+\mathbf{h}_i))\right]^2}{d_{max}^2}}, \tag{3.7}$$

with $d_{max} = \max_{\mathbf{y} \in TI} Z(y) - \min_{\mathbf{y} \in TI} Z(y)$

The third step of the algorithm consist in finding the search window $\mathcal{Y}(\mathcal{L}_i)$, traversing it (again using a random path) and testing the consecutive points $\mathbf{y}$ by computing the distance $d(\mathcal{D}(\mathbf{x}_i), \mathcal{D}(\mathbf{y}); \mathcal{L}_i)$. If the distance is lower than the predefined threshold t, then we set $Z(\mathbf{x}_i) = Z(\mathbf{y})$, otherwise a new point is tested. This step is repeated until a point with a sufficiently small distance is found or the maximal number of points (fN_{TI}) have been visited (N_{TI} stands for the number of nodes in the TI). In the latter case, the $Z(\mathbf{y})$ value corresponding to $\mathbf{y}$ yielding the smallest distance is used. DS can be summarized in the algorithmic form:

DirectSampling(n, f, t)

```
1  Load conditioning data to the simulation grid
2  N = N_SG − N_HD
3  Generate a random simulation path {x_1, x_2, ..., x_N}
4  N_TI = number of pixels in TI
5  for i = 1 to N
6       Compute X_i with n nearest neighbors
7       Compute L_i, Y(L_i)
8       min-distance = ∞
9       visited-pixels = 0
10      while cur-distance > t and visited-pixels < fN_TI
11           y = next point in Y(L_i)
12           distance = d(x_i, y; L_i)
13           if distance < min-distance
14                min-distance = distance
15                z = Z(y)
16           visited-pixels = visited-pixels + 1
17      Z(x_i) = z
```

The distance threshold parameter allows stopping immediately the search if a suitable point has been found. However, it also introduces a slight inconsistency, as it is not guaranteed that such a point exists. Therefore, the second condition must have been introduced: hitting the maximal number of scanned nodes and accepting the best candidate if no candidate satisfying the threshold has been found. DSBC simplifies the flow: no threshold parameter is used, instead the maximal scan fraction f controls the algorithm. We can replace the while loop in the line 10 of the DirectSampling algorithm, to obtain a for-loop as in the DirectSamplingBestCandidate algorithm, in the line 9. If one wants to keep the advantage of early stopping of the scan of the TI, it is possible to break from the for loop, once *distance* equal 0 has been achieved. We propose to use this option, and the tests presented in this work were performed

DirectSamplingBestCandidate(n, f, t)

```
 1  Load conditioning data to the simulation grid
 2  N = N_SG − N_HD
 3  Generate a random simulation path {x_1, x_2, ..., x_N}
 4  N_TI = number of pixels in TI
 5  for i = 1 to N
 6      Compute X_i with n nearest neighbors
 7      Compute L_i, Y(L_i)
 8      min-distance = ∞
 9      for j = 1 to f N_TI
10          y = next point in Y(L_i)
11          distance = d(x_i, y; L_i)
12          if distance < min-distance
13              min-distance = distance
14              z = Z(y)
15      Z(x_i) = z
```

3.3 Quality metrics

The connectivity function $\tau_s(\mathbf{h})$ for category s is defined as the probability that two points of the same category s, away from each other by distance $\mathbf{h}$ are connected (in the sense of belonging to the same connected component):

$$\tau_s(\mathbf{h}) = \text{Prob}\{\mathbf{x} \Longleftrightarrow \mathbf{x}+\mathbf{h} : \text{CATEGORY}(\mathbf{x}) = s, \text{CATEGORY}(\mathbf{x}+\mathbf{h}) = s\} \quad (3.8)$$

where CATEGORY is a function returning category (facies) of the argument. The indicator variogram is a variogram of the indicator variable and is computed for each category. It represents the probability that two points separated by $\mathbf{h}$ does not belong to the same category:

$$\gamma_s(\mathbf{h}) = \frac{1}{2}\mathbb{E}\left[(\mathbb{1}(\mathbf{x}+\mathbf{h}) - \mathbb{1}(\mathbf{x}))^2\right] = \frac{1}{2}\text{Prob}\{\mathbb{1}(\mathbf{x}+\mathbf{h}) \neq \mathbb{1}(\mathbf{x})\}. \quad (3.9)$$

If connectivity and indicator variogram are considered only in one direction, that is, $\mathbf{h}$ are considered always parallel to x direction, the two functions can be considered as scalar functions. In our setting, where channels go from left to right, it is justified to consider only x direction for these functions.

The statistics of the ensemble of simulations are usually compared with the TI as the reference, for example Meerschman et al. (2013) used connectivity, indicator variograms to compare how DS performed with different parameter sets in unconditional simulation case. However, due to grid size effects, the statistics of the TI differ from the statistics derived from sub-images of different sizes extracted from the TI. Therefore, we propose to compare the ensembles: ensemble of sub-images of the TI, and simulated ensembles. The comparison is achieved, for example, with Wasserstein distance, or earth mover distance. The first Wasserstein distance between two probability distributions u and v can be computed as:

$$l_1(u, v) = \int_{-\infty}^{+\infty} |U(x) - V(x)| dx, \tag{3.10}$$

where U and V are CDF of u and v respectively.

If true category values are known, it is possible to use scoring rules to assess probabilistic forecasts (Gneiting et al. 2007). For example, when conditioning data is available, cross-validation approach can be used (Juda et al. 2019) and quadratic score (2.4):

$$S(\mathbf{p}, i) = -\sum_{j=1}^{M} (\delta_{ij} - p_j)^2, \tag{3.11}$$

where $\mathbf{p}$ is probability vector, i is the true category and δ_{ij} is the Kronecker symbol. The higher the quadratic score, the better. We can also define quadratic loss (the lower, the better), simply equal to $-S(\mathbf{p}, i)$. Cross-validation score (loss) is a mean score (loss) over multiple cross-validation runs.

If true values of a continuous variable are known, continuous ranked probability score (CRPS) can be used (Gneiting et al. 2007; Gneiting and Raftery 2007):

$$CRPS(F, x) = \int_{-\infty}^{+\infty} (F(y) - \mathbb{1}(y > x))^2 dy \tag{3.12}$$

, where $F(y)$ is the CDF of the predictive distribution and x the true value. Smaller values of CRPS defined in this manner are better.

3.4 Results

Three test cases are presented, they are designed to use different features of DS, different TIs previously presented in literature, and continuous and categorical

variables. The first test case is a categorical unconditional simulation study to check the reproduction of statistics and connectivity. A pyramid version of DS is used, and it also provides an insight of how DSBC simulations change with parameters. The second is a continuous simulation with relative distances and conditioning data. It is based on filling gaps in TI, and compares probabilistic outcomes of simulation with a single true reference. The third case is a categorical simulation of a real area with trends and orientations. It emulates a real scenario, where only conditioning data is available and scores are based on 5-fold cross-validation.

Test case 1. Categorical unconditional simulations with pyramids

In this test case, a TI with four categories is used and the size of $800 \times 1\,000$ pixels representing a fluvial aquifer (Fig. 3.1). It was first presented by Jäggli et al. (2018), who used it in an inversion set-up. We will verify how DS and DSBC parameters influence the quality of simulations. To this end, sets of parameters of equal sizes are proposed for each simulation method. Each set contains 30 parameter configurations, and with each parameter set an ensemble of 40 realizations is obtained. DSBC parameter sets were simply obtained by applying the Cartesian product of a list of n and a list of f values (Tab. 3.1). DS parameter sets were obtained as union of Cartesian product corresponding to different rows of Tab. 3.2. The maximal scan fraction equals to 0.25 for all simulations. The reason for setting individual ranges for different n values is that below a certain threshold value, diminishing further this parameter has no effect, as it corresponds to setting threshold to 0, and thus running DS in DSBC mode.

For each parameter set, an ensemble of 40 realizations is generated. A realization is unconditional, the number of pixels in both directions are the same: $n_x = n_y = 200$. Multiresolution technique (gaussian pyramids) is applied with 2 pyramid levels and reduction by factor of 2 in both directions in each level (Straubhaar et al. 2020). The connectivity functions and the indicator variograms are computed only in x directions. Let h be the lag distance, and M the total number of categories. The connectivity error is given by:

$$\epsilon_c = \frac{1}{M}\frac{1}{n_x}\sum_{s=1}^{M}\sum_{h=1}^{n_x} l_1(u_s^\tau(z;h), v_s^\tau(z;h)), \tag{3.13}$$

with z going through possible connectivity values. We used $u_s^\tau(z;h)$ to indicate the distribution of connectivity values with parameter h over 40 realizations (simulations): for a constant h, the ensemble of connectivity values is retained and the probability distribution $u_s^\tau(z;h)$ deduced. The analogic distribution $v_s^\tau(z;h)$ is deduced from the ensemble of 40 images extracted from the TI of the same size $n_x \times n_y$. The indicator variogram error is computed in the same

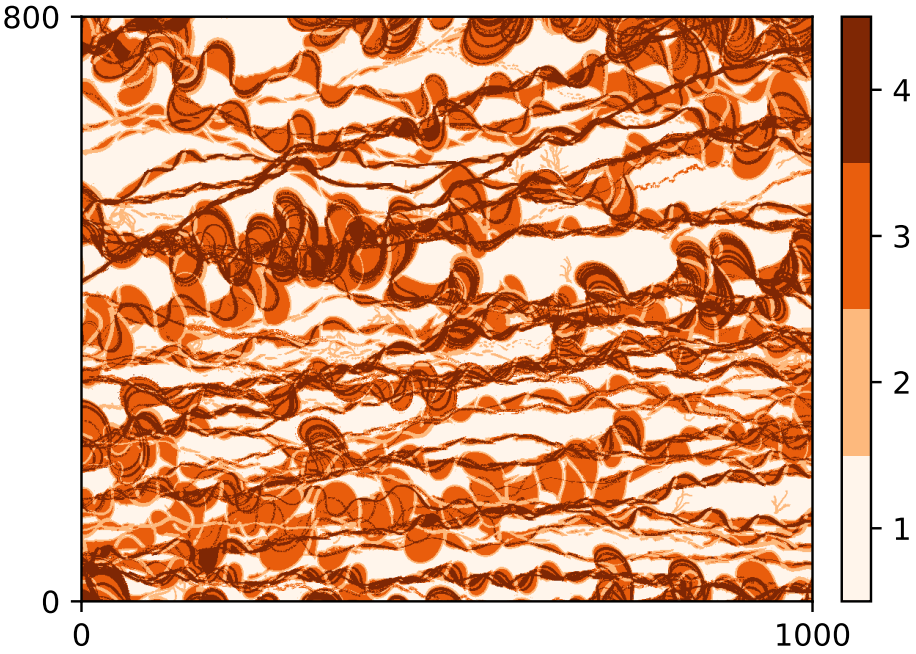

Figure 3.1: Categorical fluvial training image (Jäggli et al. 2018). The dimensions are indicated in pixels, and different categories (facies) are labelled with 1, 2, 3, 4.

n	f
8, 16, 24, 32, 64	$1/256, 1/128, 1/64, 1/32, 1/16, 1/8$

Table 3.1: DSBC parameters for Case 1. The Cartesian product of two lists gives all parameter sets.

manner, but the indicator variograms γ_s are considered in place of τ_s:

$$\epsilon_v = \frac{1}{M}\frac{1}{n_x}\sum_{s=1}^{M}\sum_{h=1}^{n_x} l_1(u_s^\gamma(z;h), v_s^\gamma(z;h)). \tag{3.14}$$

These errors are then normalized: $\varepsilon_c = \epsilon_c / \max \epsilon_c$, $\varepsilon_v = \epsilon_v / \max \epsilon_v$, and the total error is computed: $\varepsilon_t = \varepsilon_c + \varepsilon_v$. The maxima are computed over all parameter sets and over both simulation methods. The five best paremeter sets in terms of the total error for DS and DSBC are shown in Tab. 3.3. For each parameter set, an example realization is shown in Fig. 3.2 for DS and in Fig. 3.3 for DSBC. They are ordered by increasing total error. The time per simulation versus the total error are shown in Fig. 3.4A and the histogram of *time* $\times \varepsilon_t$ in Fig. 3.4B.

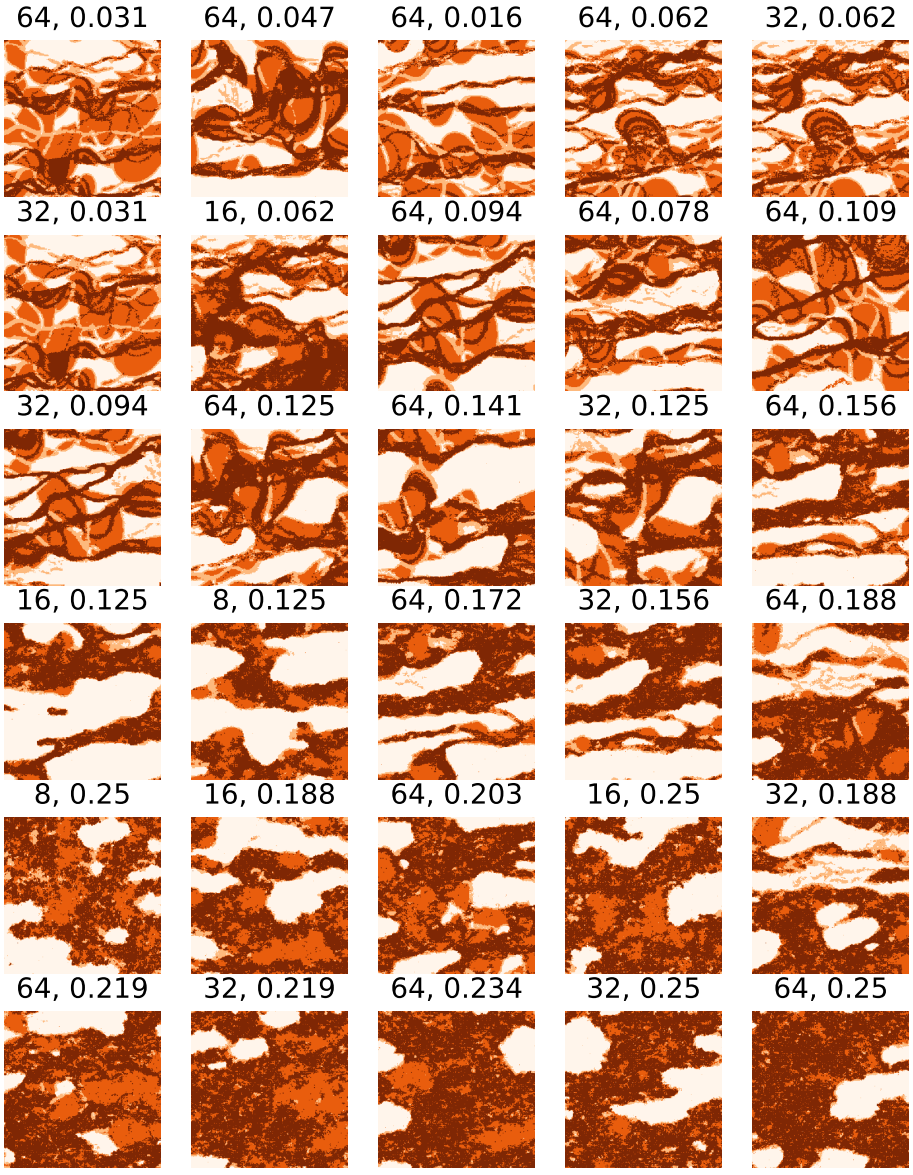

Figure 3.2: Example DS realizations ordered by descending score (increasing error). Figure titles show n and t. For the color scale refer to Fig. 3.1.

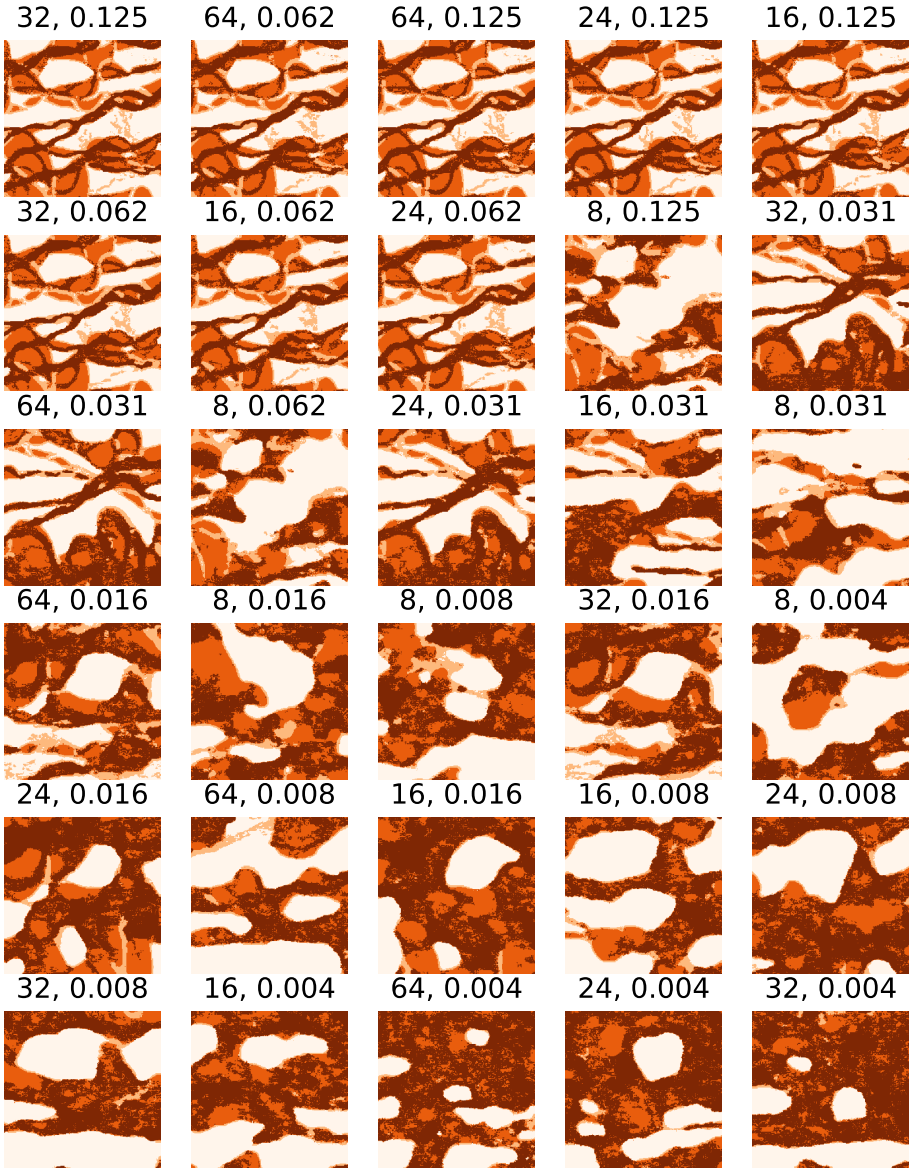

Figure 3.3: Example DS realizations ordered by descending score (increasing error). Figure titles show n and f. For the color scale refer to Fig. 3.1.TI

n	t	f
8	$1/8, 2/8$	0.25
16	$1/16, 2/16, 3/16, 4/16$	0.25
32	$1/32, 2/32, \ldots, 8/32$	0.25
64	$1/64, 2/64, \ldots, 16/64$	0.25

Table 3.2: DS parameters for Case 1. The union of the Cartesian products of two lists in each row gives all parameter sets.

	n	t	f	time	ϵ_c	ϵ_v	ε_c	ε_v	$\varepsilon_c + \varepsilon_v$
DS	64	1/32	0.25	40 s	0.032	0.012	0.186	0.094	0.279
DS	64	3/64	0.25	37 s	0.033	0.012	0.194	0.092	0.285
DS	64	1/64	0.25	43 s	0.036	0.010	0.211	0.077	0.289
DS	64	1/16	0.25	33 s	0.043	0.009	0.252	0.069	0.321
DS	32	1/16	0.25	17 s	0.043	0.009	0.251	0.072	0.323
DSBC	32	0	1/8	26 s	0.034	0.019	0.200	0.146	0.347
DSBC	64	0	1/16	30 s	0.041	0.020	0.241	0.156	0.397
DSBC	64	0	1/8	44 s	0.047	0.018	0.271	0.139	0.410
DSBC	24	0	1/8	23 s	0.046	0.019	0.266	0.153	0.419
DSBC	16	0	1/8	18 s	0.044	0.021	0.257	0.166	0.423

Table 3.3: 5 best parameter sets according to the total error for DS and DSBC.

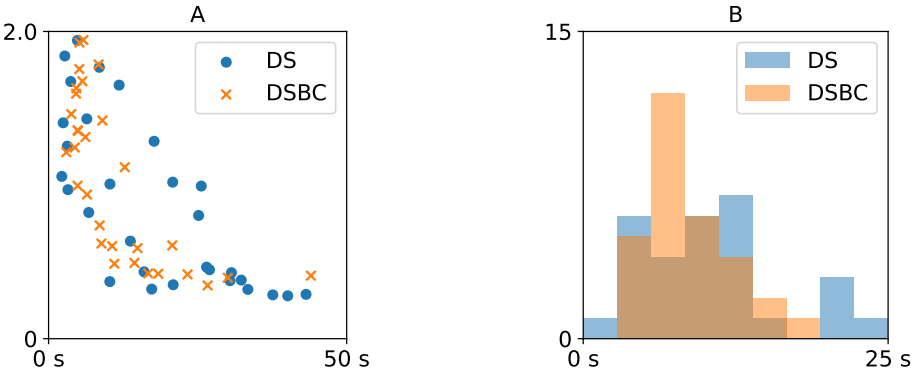

Figure 3.4: Simulation time of a single realization versus the total error ε_t (A). Histogram of *time* $\times\, \varepsilon_t$ (B).

(A) DS parameters

n	t	f
8, 16, 32, 64	0.001, 0.002, 0.005, 0.01	0.005

(B) DSBC parameters

n	f
8, 16, 32, 64	$0.001, 0.0005, 0.0002, 0.0001$

Table 3.4: DS and DSBC parameters for Case 2. The Cartesian product of the two lists for each algorithm gives all the parameter sets.

Test case 2. Continuous conditional simulations

This test case uses the methodology presented by Neven et al. (2021), which was employed to tune DS parameters for simulating the Tsanfleuron glacier bedrock geometry. The training image is the topography of Tsanfleuron glacier and the exposed bedrock after glacier retreat (Fig. 3.5) taken from Neven et al. (2021). Then, an area of 200×200 pixels is cut out from the TI, except for two perpendicular crossing lines (as an analog for GPR data), which constitute the conditioning data. DS and DSBC are used to reconstruct the missing area (by generating an ensemble of 40 realizations), different parameter sets are used (Tab. 3.4A for DS and Tab. 3.4B for DSBC). This procedure is repeated for 10 locations of bottom left corner (Tab. A.1).

An example area from the TI is depicted in Fig. 3.6A and the remaining conditioning data after cutting in Fig. 3.6A. One DS and DSBC realizations out of 40 are shown in Fig. 3.6C and D, respectively. The pointwise CRPS errors are shown as a map in Fig. 3.6E, F for DS and DSBC respectively.

The pointwise CRPS describes for each point in the simulation domain the CRPS score between the true elevation and the ensemble of elevations for this point. When these scores are averaged over all locations, the mean pointwise error (MPE) is formed. The mean volume error (MVE) describes the error of the estimation of the glacier volume. When the bedrock elevations are first averaged over all locations for single realizations, and this distribution is compared to averaged true elevation, and then CRPS score computed; the MVE score is formed. More details and rationale behind the scores were described by Neven et al. (2021).

The performance of all DS parameters is given in Tab. A.2, and DSBC in Tab. A.3 in the appendix on page 138. The tables are sorted according to ascending MVE error. The simulation time versus the total error (MPE+MVE) are shown in Fig. 3.7A and the histogram of $time \times (MPE + MVE)$ in Fig. 3.7B.

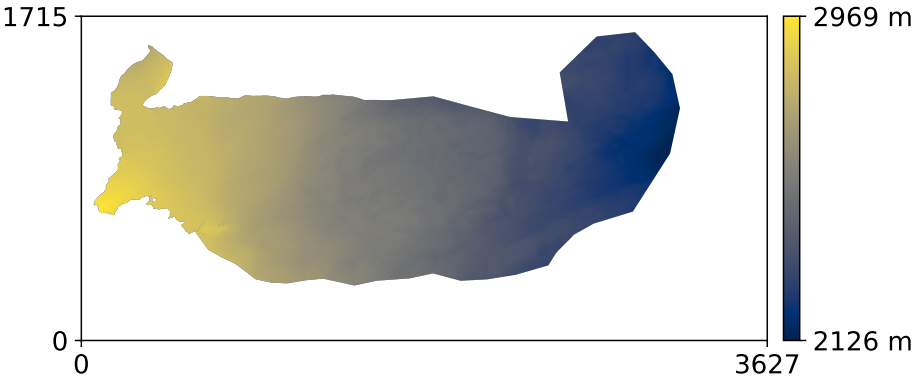

Figure 3.5: Continuous topography training image (Neven et al. 2021). The dimensions are indicated in pixels, and elevation in m.

Test case 3. Categorical multivariate simulations with cross-validation

The last test case is a synthetic but realistic case, where conditioning data is available and can be used to tune the parameters of the geostatistical method. As in a real scenario, we consider that the reality is unknown, and the only information is the conditioning data. In such a scenario, K-fold cross-validation can be used to find the best parameters of the simulation (Juda et al. 2020). This case is a simplified model of Roussillon plain (Dall'Alba et al. 2020), previously presented by Juda et al. (2020) and in Chapter 2.

The TI (Fig. 3.8C) is multivariate and has a trend (Fig. 3.8D) as secondary variable. The simulation grid has also a trend attached to it (Fig. 3.8A) and the orientation of the patterns is guided by a rotation map(Fig. 3.8B). The TI has four categories: river bed (rb), crevasse splay (s), flood plain (fp) and alluvial fan (af). We consider that 600 observation points are available (corresponding to well locations, Fig. 3.9). The 5-fold cross-validation approach with quadratic score (Juda et al. 2020) is used to compute the CV scores and their standard deviations for different parameters of DS and DSBC. The quadratic score lies between -2 and 0, with 0 corresponding to the ideal forecast. The quadratic (aka Brier) loss is the negative score, hence the lower, the better. The DS parameter set (Tab. 3.5) and the DSBC parameter set (Tab. 3.6) are very extensive, therefore only 5 best results according to CV-score are reported for each method (Tab. 3.7 for DS and Tab. 3.8 for DSBC). Example simulations by each method are in Fig. 3.10A,B for DS and DSBC respectively. The simulation plot versus CV loss (the lower, the better) and histogram of $time \times CV$ are in Fig. 3.11.

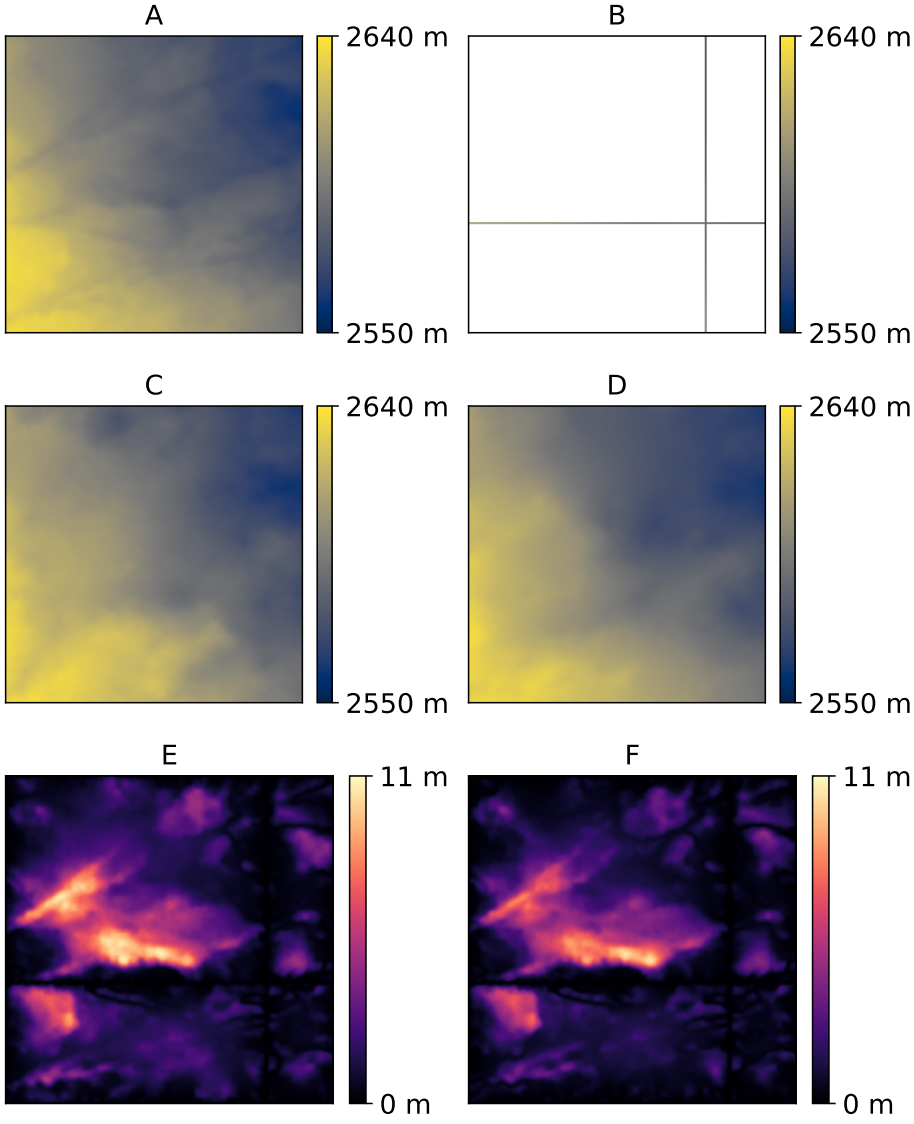

Figure 3.6: An example area to be cut out (A), corresponding conditioning data (B), example simulation by DS (C), example simulation by DSBC (D), pixelwise CRPS score map for DS ensemble (E), pixelwise CRPS score for DSBC ensemble (F).

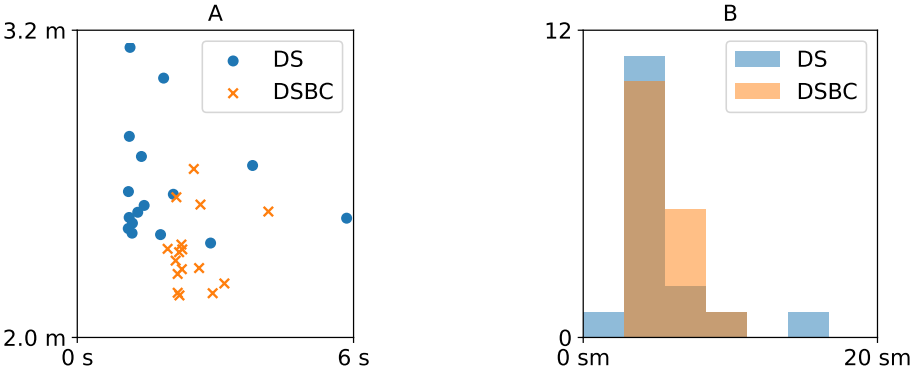

Figure 3.7: Simulation time for Case 2 of a single realization versus the total error $MVE + MPE$ (A). Histogram of $time \times (MVE + MPE)$ (B).

n	t	f
8	$1/8, 2/8$	0.1, 0.2, 0.4, 0.8
16	$2/16, 3/16, 4/16$	.1, 0.2, 0.4, 0.8
32	$1/16, 2/16, 3/16, 4/16$	.1, 0.2, 0.4, 0.8
64	$1/32, 1/16, 2/16, 3/16, 4/16$	.1, 0.2, 0.4, 0.8

Table 3.5: DS parameters for Case 3. The union of the Cartesian products of two lists in each row gives all parameter sets.

n	f
8, 16, 32, 64	.001, .002, .004, .006, .008, .01, .02, .04, .06, .08, .1, .2, .4, .6

Table 3.6: DSBC parameters for Case 3. The Cartesian product of two lists gives all parameter sets.

n	t	f	time (s)	CV-score	CV-σ
32	0.06251	0.1	4.6	-0.31	0.05
32	0.06251	0.2	4.5	-0.32	0.03
32	0.06251	0.8	7.4	-0.32	0.03
64	0.03126	0.1	4.8	-0.33	0.03
32	0.06251	0.4	4.3	-0.33	0.03

Table 3.7: 5 best DS parameter sets according to CV score.

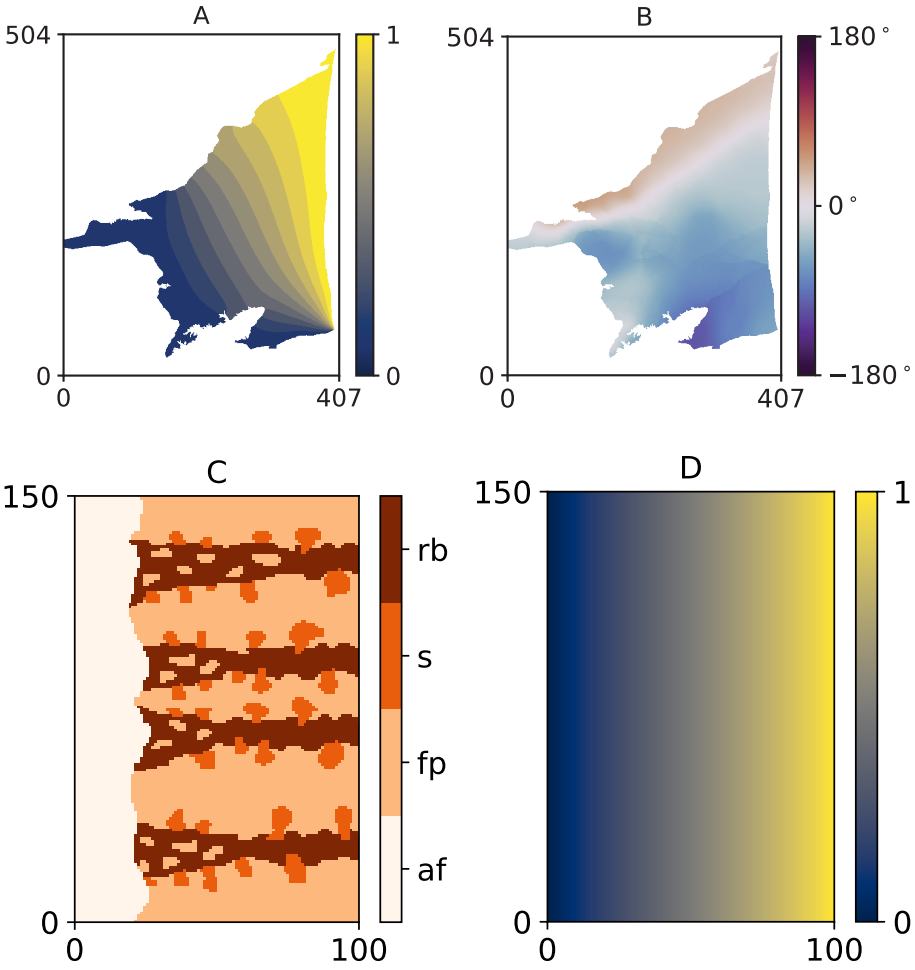

Figure 3.8: The conditioning data set containing 600 points.

n	f	time (s)	CV-score	$CV - \sigma$
16	0.02	4.3	-0.29	0.04
16	0.06	4.3	-0.29	0.03
16	0.04	4.2	-0.29	0.03
16	0.20	4.1	-0.29	0.03
16	0.08	4.2	-0.29	0.03

Table 3.8: 5 best DSBC parameter sets according to CV score.

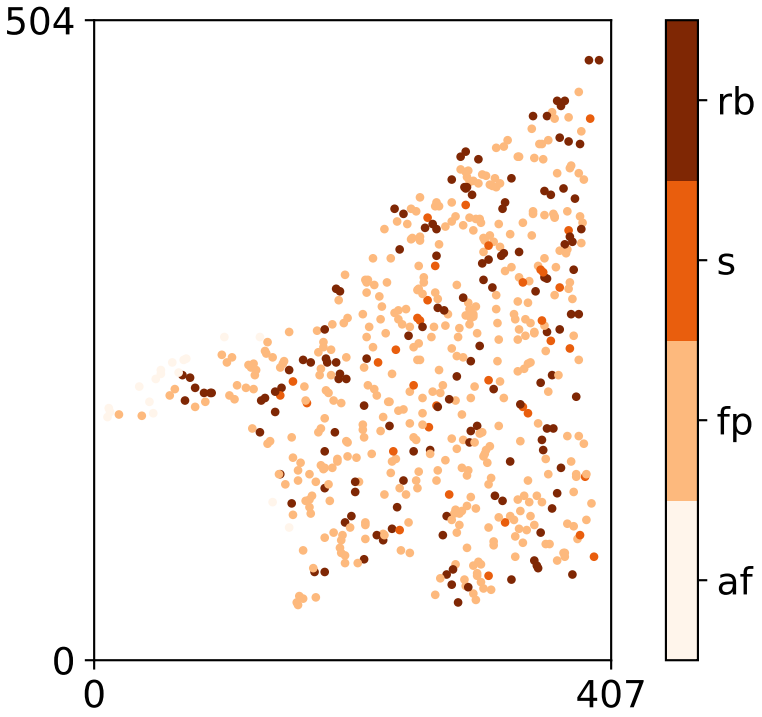

Figure 3.9: Conditioning data of Roussillon case used for cross-validation.

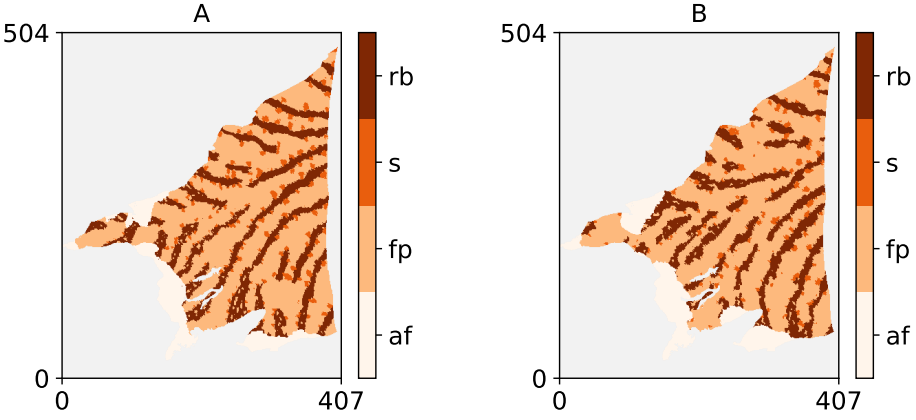

Figure 3.10: Example DS simulation (A) and example DSBC simulation (B), each performed with corresponding best parameters.

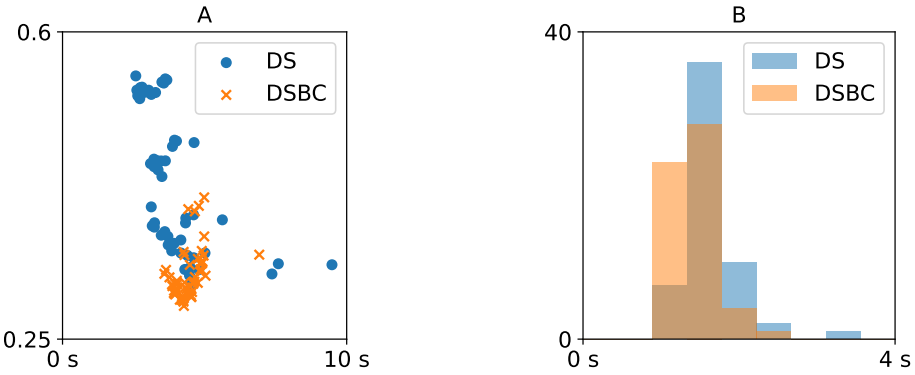

Figure 3.11: Simulation time for Case 3 of a single realization versus the CV loss (A). Histogram of $time \times CV$ (B).

3.5 Discussion and conclusion

In the first test case, DS has achieved lower errors than DSBC (Tab. 3.3) and a slightly better performance in time versus quality (Fig. 3.4A). However, DS results are more diverse: some excellent simulations (with a good time-quality trade off), but also some mediocre ones. The ideal simulations should lie in the lower left corner of Fig. 3.4A and poor ones in the upper right corner. DSBC results are more concentrated and generally good, this result concentration is confirmed by the histogram of $time \times \varepsilon_t$ values (Fig. 3.4B), where DS distribution is wider. It suggests that choosing parameter sets for testing is easier for DSBC, especially when few sets can be chosen.

In the second test case, DSBC achieved lower MVE and MPE values (Tab. A.2, A.3), and performed better in terms of total error (Fig. 3.7A), but typically needed more time per simulation. We can notice that on the scatterplot (Fig. 3.7A) DSBC points are more concentrated. DS shows more variety in terms of $time \times (MVE + MPE)$ (Fig. 3.7B).

In the third test case, DSBC yielded slightly higher cross-validation scores (Tab. 3.7, 3.8). Time analysis of simulations shows a similar, and stronger trend, with DSBC results being in general better in terms of loss and with comparable simulation time (Fig. 3.11A). Moreover, the distribution of performance indicator is clearly better for DSBC with good performance being much more common than for DS (Fig. 3.11B).

In summary, in this chapter we have formulated and tested the DSBC geostatistical method. It is a simplified version of DS algorithm. DSBC has only two major algorithmic parameters, as compared to three for the DS method. It simplifies parameter tuning, while retains all the advantages of DS algorithm. The

three test cases that we studied confirmed that the quality and performance of DSBC is similar to DS. We found that DSBC is more likely to produce good simulations when used with different parameters. DSBC can also be implemented as a parameter tuning strategy for the existing DS implementations. Therefore, DSBC is a simple and efficient default manner for choosing DS parameters. Our recommendation is to start by applying the DSBC strategy and if the results are not satisfactory, then one can refine more parameters using the complete DS algorithm as it has more tuning capabilities. Further research should investigate the possibility of an efficient implementation of DSBC, for instance for GPUs. The fact that the loop corresponding to the scan in the TI is simplified, might lead to an optimal implementation tailored specifically for DSBC. A similar idea of performing full TI scan (removing distance threshold parameter and fraction scan, but introducing k parameter for "k-sampling") and specializing its implementation using FFT was proposed by Gravey and Mariethoz (2020). Their QuickSampling algorithm is statistically similar to DSBC (with $f = 1/k$) and computationally efficient, but it is not clear how it could include advanced DS features. Therefore, combining the flexibility of DSBC and computational advantages of QuickSampling would be highly desirable.

Chapter 4

An attempt to boost posterior population expansion using fast machine learning algorithms[1]

Abstract

In hydrogeology, inverse techniques have become indispensable to characterize subsurface parameters and their uncertainty. When modeling heterogeneous, geologically realistic discrete model spaces, such as categorical fields, Monte Carlo methods are needed to properly sample the solution space. Inversion algorithms use a forward operator, such as a numerical groundwater solver. The forward operator often represents the bottleneck for the high computational cost of the Monte Carlo sampling schemes. Even if efficient sampling methods (for example Posterior Population Expansion, PoPEx) have been developed, they need significant computing resources. It is therefore desirable to speed up such methods. As only a few models generated by the sampler have a significant likelihood, we propose to predict the significance of generated models by means of machine learning. Only models labeled as significant are passed to the forward solver, otherwise, they are rejected.

This work compares the performance of AdaBoost, Random Forest, and convolutional neural network as classifiers integrated with the PoPEx framework. During initial iterations of the algorithm, the forward solver is always executed and subsurface models along with the likelihoods are stored. Then, the machine learning schemes are trained on the available data. We demonstrate the technique using a simulation of a tracer test in a fluvial aquifer. The geology is modeled by the multiple-point statistical approach, the field contains four geological facies, with associated permeability, porosity, and specific storage

[1]This chapter was published as: Juda P and Renard P (2021) An Attempt to Boost Posterior Population Expansion Using Fast Machine Learning Algorithms. *Front. Artif. Intell.* 4:624629. doi:10.3389/frai.2021.624629

values. MODFLOW is used for groundwater flow and transport simulation. The solution of the inverse problem is used to estimate the 10 days protection zone around the pumping well. The estimated speed-up with Random Forest was higher than with AdaBoost and convolutional neural network. To validate the approach, computing times of inversion without and with Random Forest were computed and the error against the reference solution was calculated. For the same mean error, PoPEx accelerated with Random Forest achieved a speed-up rate of up to 2 with respect to the standard PoPEx.

4.1 Introduction

Groundwater flow and contaminant transport in aquifers depend on subsurface parameters such as permeability, specific storage, or porosity. Due to the lack of their direct measurements and heterogeneity, solving inverse problem is essential to make reliable predictions of groundwater flow and crucial for water resources management.

The inverse problem consists in deducing the subsurface parameters given state variables measured in the field, such as hydraulic heads or tracer concentrations. The physical problem leading from parameters to state variables is called the forward problem; it often involves solving partial differential equations. While the forward problem has a unique solution, the inverse problem is usually ill-posed, with non-unique and unstable solution if framed in a deterministic manner. The inverse problem is especially difficult when dealing with highly heterogeneous parameter fields or categorical fields. Therefore, methods for solving the inverse problem in hydrogeology (and more broadly in geophysics) have been a topic of extensive research (Linde et al. 2015; Zhou et al. 2014). If defined in a probabilistic manner, the inverse problem is no longer ill-posed and the solution always exists (Tarantola 2005). When formulated in a Bayesian framework, it needs defining prior knowledge which allows to properly account for subsurface heterogeneity. The associated difficulty lies in estimating the probability density over a non-linear space of parameters and requires using a Monte Carlo method for generating many parameter fields and forward model runs. Depending on the physical problem, the forward model can be computationally expensive (especially true for transient groundwater flow or contaminant transport models); the inversion would typically require long runs and using high-performance computing resources.

Machine learning, including deep learning, has gained momentum in water research (Shen et al. 2018), and can be used to improve the efficiency of the inverse methods (Marçais and de Dreuzy 2017). There are two main areas of application of machine learning related to the inverse problem: first, using machine learning techniques for modeling the prior knowledge; second, emulating the forward problem.

Concerning the generation of samples from the prior distribution, generative

adversarial networks (GAN) are interesting because of their ability to generate geologically realistic models. Despite significant training times, they are attractive for the fast generation of fields and because they offer a low dimensional parametrization, which allows an efficient exploration of model space (Chan and Elsheikh 2020; Laloy et al. 2017). As a consequence, GANs have been successfully used in Markov Chain Monte Carlo algorithms for solving groundwater inversion problems (Laloy et al. 2018, 2017). Due to the possibility of computing gradients in GAN models space, deterministic inversion using GANs is also possible but can be hindered by the non-linearities of the problem (Laloy et al. 2019). Simpler models, such as Support Vector Machines (SVM) were also used to construct informative geological prior to constrain sampling for realistic reservoir models (Arnold et al. 2019).

Concerning the emulation of the forward problem, machine learning can also be used. For example, Tripathy and Bilionis (2018) used deep neural network to construct a surrogate for a stochastic partial differential equations and employed it in the context of high dimensional uncertainty quantification. Laloy and Jacques (2019) tested machine learning algorithms (including deep neural nets, Gaussian processes, polynomial chaos expansion) to emulate reactive transport model and found that deep neural networks perform reasonably well in the context of quantifying uncertainty. Dagasan et al. (2020) showed how GAN can emulate steady-state flow solver and used it in the posterior population expansion algorithm (Jäggli et al. 2018), showing that the results of the inversion were of similar quality as those obtained using the numerical flow solver.

All these approaches rely on machine learning techniques built specifically for a problem at hand and replacing state-of-the-art methods for either modeling the prior, or emulating the forward problem.

In this paper, we propose a slightly different approach based on a machine learning classifier, and not substituting the prior geostatistical model, nor the forward solver. We use the posterior population expansion method (PoPEx) to invert a categorical field and our aim is to accelerate the convergence of the algorithm. During the Monte Carlo exploration of the model space, models are generated along with the forward operator results. These data are then used to train a classifier, which predicts if models would have high or low likelihood and whether they would contribute much to the posterior distribution. Then, the classifier can decide if a forward operator should be called (when the parameter field is predicted to be favorable) or if a model can be discarded, saving computational time. This approach is generic, as it does not rely on a specific geostatistical prior model, neither on a forward solver type. It is similar in spirit to the approach of building a surrogate or emulated model (like Dagasan et al. (2020) and Laloy and Jacques (2019)), but instead of reproducing the forward response, it predicts if the response would match well with the observed data. A similar idea was proposed by Demyanov et al. (2010), where SVM classifier

was applied to separate high likelihood models while stochastically sampling parameters space for reservoir predictions.

To test the efficiency of the proposed method, We consider an alluvial aquifer. Its geological heterogeneity is modeled using multiple-point statistics. The inverse problem consists in interpreting one tracer test. Once the geological heterogeneity and its remaining uncertainty is identified, the resulting distribution of geological fields is used to predict the 10-days capture zone. This problem is used since it is a frequent question in applied hydrogeology. One has to interpret tracer test data to then delineate protections zones around future drinking water production wells. Therefore, in addition to the analysis of the efficiency of the inverse method, we check also the quality of this prediction as compared to the reference.

4.2 Methods

In this section, we present a brief review of the posterior population expansion algorithm, and the machine learning schemes with their architectures used in this study. Finally, we explain how the schemes are used to accelerate PoPEx algorithm.

Inverse problem and PoPEx algorithm

PoPEx has been previously introduced and presented in detail in Jäggli et al. (2017) and Jäggli et al. (2018). It solves the inverse problem in a probabilistic manner. First, we explain how the problem is framed; then, we review the algorithms and its parameters; finally, we show how the solution can be used to generate a prediction.

Probabilistic formulation

We will consider that $\mathbf{d}_{obs} \in \mathbb{R}^n$ is a data set obtained from an experiment with n defining the number of data points. The data are physical state variables, for example: hydraulic head, contaminant concentrations, or flow rates. Let $\mathbf{g} : \mathcal{M} \to \mathbb{R}^n$ be the forward operator, mapping from the model space $\mathcal{M}$ to the data space $\mathbb{R}^n$. The model space defines the set of physical parameters, which fully describe the system (for example subsurface parameters) and allows to solve the forward problem (for example a groundwater flow problem). The forward operator generates the observable data given the model. In our case, the forward operator is a groundwater flow and transport model, and the model space is a set of all possible geological realizations of the subsurface, which map to permeability, porosity and specific storage.

The probabilistic solution of the inverse problem is given by (Tarantola 2005):

$$\sigma(\mathbf{m}) = c\rho(\mathbf{m})L(\mathbf{m}), \quad \mathbf{m} \in \mathcal{M}$$

with σ the posterior probability density, ρ the prior probability density, L the likelihood, and c a normalization constant. The prior probability density is defined by expert knowledge about the parameter field. For example, it can constrain the type of geology which is considered. The likelihood evaluates the mismatch between the data and simulated values (output of the forward operator).

PoPEx algorithm

In the following, we provide a rapid and brief overview of PoPEx. The posterior population expansion algorithm (Jäggli et al. 2018) is a modified adaptive importance sampler. It iteratively expands a sampled model set and learns the underlying probability distribution. It is designed for solving inverse problems when the space $\mathcal{M}$ contains categorical models. It requires a conditional geostatistical tool (e.g. a geostatistical algorithm which can generate a new model $\mathbf{m}$ from space $\mathcal{M}$ given conditioning points); and a forward solver which computes the response values at the data points and a likelihood function for a given model. At each iteration k it expands the sampled model set and uses all previously generated models and their likelihoods to define the conditioning set for the next model $\mathbf{m}_{k+1}$. The number of conditioning points at every iteration is uniformly drawn from $\{0, \ldots, n_c\}$, where n_c is the maximal number of conditioning points — a parameter specified by the user. Then, the new model is generated by the geostatistical tool honoring the imposed conditioning data, the response is computed by the forward solver, and the likelihood of the model is evaluated. The PoPEx algorithm stops after a specified number of iterations N given by the user.

The choice of the conditioning data locations is guided by the Kullback-Leibler divergence (KLD) map computed between two discrete probability distributions P and Q on the probability space $\mathcal{X}$ and at each point in the domain:

$$D(P^k||Q) = \sum_{x \in \mathcal{X}} P^k(x) \log\left(\frac{P^k(x)}{Q(x)}\right). \tag{4.1}$$

The discrete case is considered and $\mathcal{X}$ is constituted of the ensemble of s possible categories (eg. geological facies), $Q = \{q_1, \ldots, q_s\}$ corresponds to the prior probability maps for each category, $P^k = \{p_1^k, \ldots, p_s^k\}$ corresponds to the maps of facies probabilities but weighted by the normalized likelihoods estimated at iteration k:

$$\tilde{L}(m_j) = L(m_j)/(\sum_{r=1}^{k} L(m_r)).$$

The higher the KLD for a given point, the more likely it is chosen as a conditioning data location. Once conditioning point locations are determined, their values are sampled from the local P distribution.

Prediction

Typically, solutions of the inverse problem are used to generate some predictions. Let f be the function mapping from models to predictions, and μ the expected value of some quantity of interest (e.g. prediction of hydraulic head, capture zone of a well):

$$\mu = \int_{\mathcal{M}} \sigma(\mathbf{m}) f(\mathbf{m}) d\mathbf{m}. \tag{4.2}$$

PoPEx uses the set of generated models to approximate (4.2):

$$\widehat{\mu} = \sum_{i=1}^{N} \widehat{w}_i f(\mathbf{m}_i), \tag{4.3}$$

with $\widehat{w}_i$ a normalized *corrected* weight of the model i. The *corrected* weights are computed from the sampling weights w_i:

$$\widehat{w}_i = \frac{w_i^\alpha}{\sum_{i=j}^{N} w_j^\alpha}, \tag{4.4}$$

with α the correcting factor. The weights w_i are given by:

$$w_i = L(\mathbf{m}_i) \frac{\rho(\mathbf{m}_i)}{\phi_i(\mathbf{m}_i)}, \tag{4.5}$$

where ρ is the prior distribution function and ϕ_i is the sampling distribution used at the iteration i. The weights need to be adjusted, as with large model spaces the distribution $W^N = \{w_1, \ldots, w_N\}$ can be dominated by few very large samples. The effective sample size n_e provides a measure of the skewness of the distribution:

$$n_e(W^N) = \frac{(\sum_{i=1}^{N} w_i)^2}{\sum_{i=1}^{N} w_i^2}. \tag{4.6}$$

Correcting the weights consists of finding the factor $\alpha \in (0, 1]$ as close (or equal) to 1 such that the effective number of weights of the set $\{w_1, \ldots, w_N\}$ is at least l_0, with l_0 specified by the user.

Machine learning methods

Binary classification is a supervised learning task that has been extensively studied in the context of predictive data analytics (Kelleher et al. 2015), statistical learning (Hastie et al. 2009), and deep learning (Goodfellow et al. 2016). It consists in predicting the binary label (binary class, encoded as 0 or 1) given

the input data (features). If X is the input (a feature vector, containing categorical or continuous variables), a binary classifier $\hat{f}$ maps X to the binary label $\hat{y}$: $\hat{f}(X) = \hat{y}$ with $\hat{y} \in \{0, 1\}$. The classifier must be fitted to known data with assigned labels, and then it can be used to generate predictions on previously unseen data. In this work, we considered three classifiers: AdaBoost, Random Forest and Convolutional neural network (CNN). Here, we briefly introduce these methods and the cross-validation technique which was used to tune parameters of the classifiers and select the best algorithm. We refer to Hastie et al. (2009) for introductions about AdaBoost, Kelleher et al. (2015) about decision trees, Breiman (2001) about Random Forest, and Goodfellow et al. (2016) about CNN and deep learning in general.

Classifiers

AdaBoost is a boosting method (Freund and Schapire 1997) which produces a sequence of weak classifiers. The weak classifiers are iteratively fitted to the data with weights modified at each step. More weight is given to previously missclassified samples (initially weights are equal). The final prediction of the AdaBoost algorithm is a majority vote of all classifiers. Typically the weak classifiers used in AdaBoost are shallow decision trees. The error on the training sample is the average of the fraction of misslassified samples. The fitting is finished when perfect fit is achieved or when the total number of estimators has been reached. Random Forests (Breiman 2001), similarly to AdaBoost, rely on ensembles of decision trees. The decision trees are typically deeper than in AdaBoost and they are trained on data randomly sampled from the input. The sampling distribution from which the training data is drawn is the same for all trees in the forest. The main parameters of the method are the tree depth and the number of trees that form the forest. Breiman (2001) claimed that random forests compare favorably to AdaBoost, yielding similar errors and being more robust with respect to noise. For gridded inputs, convolutional neural networks (LeCun et al. 1989) proved to be effective. Convolutional neural networks are a special type of neural nets (Goodfellow et al. 2016), composed of convolutional layers, combined with activation, pooling and dense layers. They are especially suitable for object recognition and are state-of-the-art classifiers for working with images.

Cross-validation

Cross-validation is a technique which allows to estimate the error (or a score) of the classifier on the unseen data. In K-fold cross-validation the whole training data is divided into K subsets and K iterations are performed. In each iteration $iter = 1, \ldots, K$, the $iter$ subset in a row is removed from the data to form the validation set. The rest of the data becomes the training set. The classifier is fitted to the training set and the error (or a score) is computed using the validation set: predictions are made on the validation set and they are compared to true values. Then, gathering the output of K iterations, one

can compute the statistics of the error or the score function. Kohavi (1995) argued that the best choice is $K = 5$ or $K = 10$.

Performance measures

We introduce here the commonly used performance measures for binary classifiers, which are based on the confusion matrix. Let us suppose that the classification results in TP true positives, TN true negatives, FP false positives and FN false negatives. Precision is defined as follows:

$$\text{precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}, \tag{4.7}$$

it describes how confident we can be that the predicted positive instance is correct. Recall is given by:

$$\text{recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})}, \tag{4.8}$$

and it describes the ability of the model to find all positive instances. The harmonic mean of precision and recall is F score (or F_1 score) and it is generalized by F_β score (Rijsbergen 1979):

$$F_\beta = \left(1 + \beta^2\right) \frac{\text{precision} \cdot \text{recall}}{\beta^2 \text{ precision} + \text{recall}}, \tag{4.9}$$

where β parameter describes how more recall is important over precision. For $\beta = 1$, the F_β falls back on standard F-score which is a harmonic mean of precision and recall.

Accelerating PoPEx

At each PoPEx iteration i, a new model $\mathbf{m}_i$ is generated. Instead of feeding it directly to the forward solver, a learning scheme can predict if the model's likelihood is significant enough to contribute to the solution of the inverse problem. If the model is not especially useful, it can be discarded; its likelihood is set to 0. Otherwise, if the learning scheme predicts that the model is good enough, the forward solver is called and the exact likelihood of the model is computed. In this way, expensive forward solver is not called for every model. Then, PoPEx proceeds to draw another model and the procedure is repeated. The discarded models do not contribute to the solution; even if the learning scheme makes a mistake, it will not bias the results. Discarding useful models or marking low-likelihood models as good, would only slow down the convergence of PoPEx but would not introduce incorrect likelihood estimations.

Application of learning scheme

The learning scheme needs to be trained before being used with PoPEx. To this end, a sufficient dataset of pairs $(\mathbf{m}_i, L(\mathbf{m}_i))$ must be generated. To achieve

this, we first run PoPEx in the normal mode, for a specified number n_t of iterations, evaluating all models using the forward solver.

Then, the machine learning scheme is trained using the pairs $\{(X_i, y_i), i = 1 \dots n_t\}$ with X_i the input data and y_i the classification labels. X_i can be models $\mathbf{m}_i$ or some data obtained by transforming $\mathbf{m}_i$, for example collection indicator variables, or physical properties (porosity, conductivity) derived from $\mathbf{m}_i$. $y_i \in \{0, 1\}$ is a binary label describing if a model is useful (0 meaning an insignificant model with low likelihood and 1 – important model with high likelihood). We propose to sort all available likelihoods: $\{L(\mathbf{m}_i), i = 1 \dots n_t\}$, define a ratio $r \in (0, 0.5)$ and set $y_i = 1$ if $L(\mathbf{m}_i)$ is among the rn_t highest likelihoods, and $y_i = 0$ otherwise.

Once the learning scheme has been trained, the classifier is used to classify each new model $\mathbf{m}_j, j > n_t$. Let $\hat{y}_i$ be the prediction of the true label y_i. If $\hat{y}_i = 1$, the model $\mathbf{m}_j$ is fed to the forward solver and its true likelihood is evaluated. If $\hat{y}_i = 0$, we set $L(\mathbf{m}_j) = 0$ and the forward solver is not called. The true likelihood of the model remains unknown and the model does not contribute to the solution of the inverse problem, nor influences the sampling scheme of PoPEx.

This methodology is motivated by the fact that often only very few models have high likelihood (Jäggli et al. 2018). Correcting of the prediction weights was designed to deal with this problem. Therefore, it is justified to discard models with low likelihood, as they would not influence the PoPEx predictions anyway. Applying a perfect classifier in this way, should produce practically the same results as the original PoPEx algorithm.

A classifier yields false positives and false negatives. False positives (e.g. classifying low-likelihood models as useful) cause that an uninteresting model is being fed to the forward solver. A classifier producing a lot of false positives would have a weaker speed-up abilities. False negatives are more problematic, as it means that good models are discarded. A lot of false negatives would slow down the convergence of PoPEx, as the sampling scheme does not benefit from updating KLD maps and learns more slowly.

Speed-up score and evaluation metrics

Given the training set for the classifier, it is useful to estimate the possible speed-up when applying it in the PoPEx scheme. We suppose here that we want to achieve an inversion result equivalent (in the sense of prediction error) to running plain (standard) PoPEx for N iterations. Let us compare the total cost of generating an ensemble of N PoPEx models and the total cost of generating a bigger ensemble with the machine learning scheme with the equivalent number of significant models. More models are needed to account for the fact that due to the imperfect learning scheme, some potentially good models are

discarded. To simplify the calculation, we suppose that generated new models have the same chance to be good, and we neglect the fact that using ML scheme influences PoPEx sampling and the quality of generated models.

Let c_m be the computational cost of generating a model $\mathbf{m}$, c_g the cost of running the forward problem solver and we set $c = c_m/c_g$. We assume that the forward solver is more expensive than running the geostatistical model, thus $c < 1$. The total computational cost for generating an ensemble with N elements without ML is $N \cdot (c_m + c_g)$. Suppose that out of N models, ML scheme generates TP true positives ($\hat{y}_i = 1$ and $y_i = 1$), TN true negatives ($\hat{y}_i = 0$ and $y_i = 0$), FP false positives ($\hat{y}_i = 1$ and $y_i = 0$) and FN false negatives ($\hat{y}_i = 0$ and $y_i = 1$). More models need to be generated to account for the fact that some positives are not detected, so the total number of models N must be multiplied by $(\mathrm{TP} + \mathrm{FN})/\mathrm{TP}$. All true and false positives require the forward solver, thus the total cost with ML will be:

$$\frac{\mathrm{TP} + \mathrm{FN}}{\mathrm{TP}} \left(c_m N + c_g(\mathrm{TP} + \mathrm{FP})\right).$$

We propose to use the ratio of the total computational cost without ML divided by the cost when using ML scheme as a speed-up estimator. Let us call it *s-score* ($\mathrm{s}_{\mathrm{score}}$):

$$\mathrm{s}_{\mathrm{score}} = \frac{N \cdot (c_m + c_g)}{\left[c_m N + c_g(\mathrm{TP} + \mathrm{FP})\right] \cdot (\mathrm{TP} + \mathrm{FN}) / \mathrm{TP}}$$

Using the definitions for precision and recall (4.7,4.8) and the ratio of "good" models $\mathrm{TP} + \mathrm{FN} = rN$, we obtain:

$$\mathrm{s}_{\mathrm{score}} = \frac{N(c_m + c_g)}{\frac{1}{\mathrm{recall}} c_m N + c_g \frac{1}{\mathrm{precision}} rN} = \frac{1 + c}{\frac{c}{\mathrm{recall}} + \frac{r}{\mathrm{precision}}}. \tag{4.10}$$

When generating a model is significantly cheaper than running the forward solver, $c \to 0$ and we obtain a convenient approximation:

$$\mathrm{s}_{\mathrm{score}} \approx \mathrm{precision}/r, \tag{4.11}$$

useful to evaluate rapidly if a learning scheme is potentially a good candidate for accelerating PoPEx sampling. If c values are greater than 1, the machine learning scheme is not interesting to apply, even if the recall is high. Smaller r would yield potentially large *s-score* (if the precision is high). The *s-score* can also be expressed in terms of F_β score (4.9):

$$\mathrm{s}_{\mathrm{score}} = \frac{1 + c}{r + c} F_\beta,$$

where $\beta^2 = c/r$. If $c < r$, β is lower than 1 (and close to 0), which attributes more importance to precision than recall.

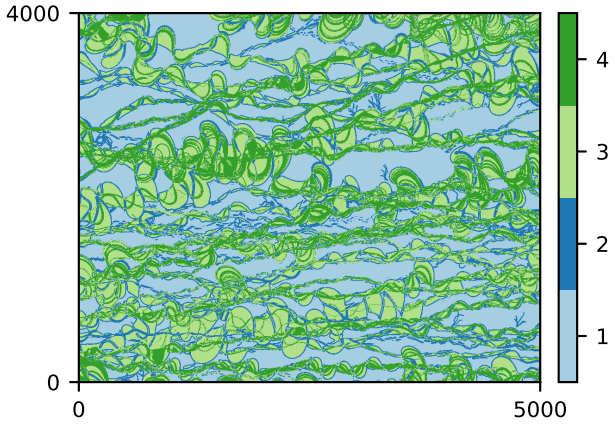

Figure 4.1: Training image used for generating the synthetic reality and in the inversion process.

4.3 Test Case

We consider a 2D synthetic reference field representing a fluvial aquifer. The data used for the inversion is the synthetic tracer breakthrough curve recorded at the pumping well. The reference probabilistic solution for the inverse problem is obtained after 40 000 PoPEx iterations, it will be used to assess the solutions generated using the learning schemes.

Synthetic model of a fluvial aquifer

The 2D geology is modelled by multiple-point statistics (MPS), which allows to account for subsurface heterogeneity and represent categorical fields (Mariethoz and Caers 2015). The MPS algorithm requires a training image (TI), an example of the field which is used as a spatial pattern database. Here, we consider a TI composed of 4 geological facies, representing a fluvial aquifer (Fig. 4.1), and first published by Jäggli et al. (2018). The TI has a dimension of 1 000 pixels by 800 pixels with cell size of 5 m x 5 m. To obtain the synthetic reference field (Fig. 4.2A), we performed a Direct Sampling (Mariethoz et al. 2010c) simulation on a regular grid representing 500 m by 500 m area (100x100 pixels). The DeeSse code with multi-resolution capabilities (Straubhaar et al. 2020) was used with the following parameters: 2 pyramid levels (with a pyramid for each indicator variable) with reduction factor 2 at each level and each direction (x, y); search neighborhood radius: 40 in each direction, number of neighboring nodes: 60, distance threshold: 0.01, maximal scan fraction: 0.04. The reference realization was generated with seed value of 201 913. Moreover, points at $x = 374.5$ m, $y = 249.5$ m and $x = 124.5$ m, $y = 249.5$ m are considered to have known facies of category 4, in other words conditioning data is imposed in these two locations with value 4.

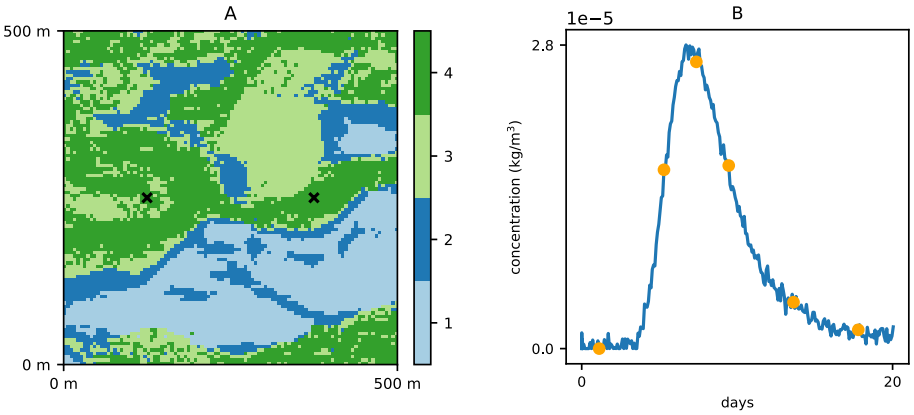

Figure 4.2: The reference field **(A)** used as the synthetic reality (considered unknown) and the corresponding concentration curve **(B)**.

Groundwater flow and transport

The aquifer is a confined fluvial aquifer with thickness of 10 m and modeled as a single layer. Each geological facies has a uniquely defined physical properties (hydraulic conductivity, porosity and specific storage) and can be interpreted as different rock type: silt, fine sand, coarse sand and gravel (Tab. 4.1). The maps of conductivity, porosity and specific storage on a 100x100 regular grid obtained by means of DeeSse simulations are refined by factor of 5 in each direction, resulting in a 500 m x 500 m model with cell size 1 m by 1 m. The value of a parameter at each location is equal to its value in the parent cell.

A pumping well is placed at $x = 374.5$ m, $y = 249.5$ m and it pumps water with constant rate of $0.07\,\mathrm{m}^3/\mathrm{s}$. The left (west) boundary is at constant head 0.5 m, the right boundary (east) at constant head 0 m and the hydraulic heads at top (north) and bottom (south) boundaries are linearly interpolated between 0.5 m and 0 m. The groundwater flow and transport model was implemented using the flopy python package Bakker et al. 2016. The steady-state solution of the flow problem is used as initial condition for a transient groundwater flow and transport simulation of a tracer test. The injection well is placed at $x = 124.5$ m, $y = 249.5$ m. The tracer is injected at a constant concentration of $1\,\mathrm{kg}/\mathrm{m}^3$ with a constant injection rate of $1\,\mathrm{m}^3/\mathrm{h}$, so that a total of $1\,\mathrm{m}^3$ water is injected during 1 h. Afterwards, the injection is stopped.

The total simulation time is 1 h plus 20 days, discretized in 36 time-steps during initial 1 h and 240 time-steps during the rest of the simulation time. During this period, the flow is modeled in transient regime to account for the perturbation due to the injection. The transport simulation is done with the following parameters: diffusion coefficient of $1 \times 10^{-9}\,\mathrm{m}^2/\mathrm{s}$; longitudinal dispersivity of 4 m and transversal dispersivity of 0.4 m. The concentration curve at the pump-

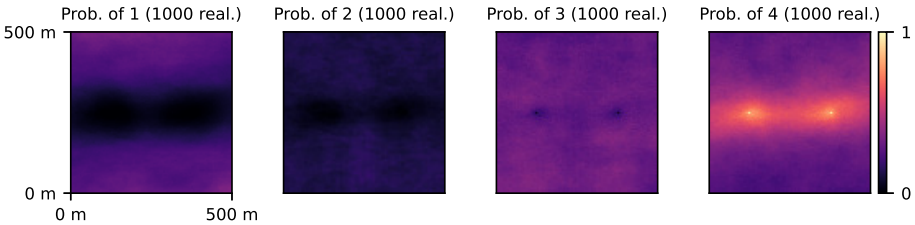

Figure 4.3: Prior probability maps associated with different geological facies (labelled 1, 2, 3, 4).

ing well was recorded and random Gaussian noise with mean 0 and standard deviation $\sigma_L = 0.05 \times 10^{-6}\,\mathrm{kg/m^3}$ was added to emulate measurement error (Fig. 4.2B).

Reference solution of the inverse problem

In this scenario, the prior distribution of the geological fields is modeled using the MPS technique with the same parameters as those used to generate the reference realization. Two conditioning points are imposed: the facies 4 (permeable, gravel) at the pumping well and injection well. A prior ensemble of 1 000 realizations was generated and facies probability maps (Fig. 4.3) were used as input for the PoPEx algorithm (prob. Q in 4.1).

The likelihood for this problem should be written as proportional to

$$\exp\left(-1/(2\sigma_L^2)\sum_{i=1}^{n}\left(\mathbf{g}_i(\mathbf{m}) - \mathbf{d}_i^{obs}\right)^2\right) \tag{4.12}$$

with σ_L the standard deviation of measurement error, $\mathbf{g}(\mathbf{m})$ the simulated values of concentration of the model $\mathbf{m}$, and $\mathbf{d}^{obs}$ the vector of measurements. However, when using this complete formula, the likelihood values are zero for most realizations (considering the fact that numerical representation of floating point numbers has a finite number of digits) and PoPEx converges very slowly (Jäggli et al. 2018). Therefore, to avoid this numerical issue, and following Jäggli et al. 2018, we considered a reduced data set including only six time steps (i) to estimate the likelihood. The *sampled* likelihood formula was:

$$\exp\left(-1/(2\sigma_L^2)\sum_{i\in I}\left(\mathbf{g}_i(\mathbf{m}) - \mathbf{d}_i^{obs}\right)^2\right) \tag{4.13}$$

with $I = \{50, 100, 125, 150, 200, 250\}$, which corresponds to the time points {29 h, 129 h, 179 h, 229 h, 329 h, 429 h}, depicted in Fig. 4.2.

PoPEx was run for 40 000 iterations with $l_0 = 100$ and $n_c = 10$. The 10 days groundwater capture zone (van Leeuwen et al. 1998) was predicted with the

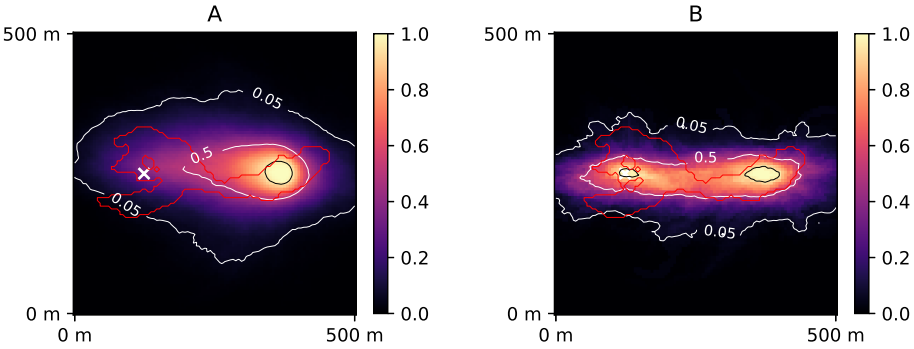

Figure 4.4: Prior (**A**) and posterior (**B**) probability maps of 10 days zone. The red contour corresponds to the reference 10 days zone (simulated using the synthetic reality; considered unknown). The white cross represents the injection well. The black contour corresponds to 95 % confidence. The posterior was obtained after 40 000 PoPEx iterations.

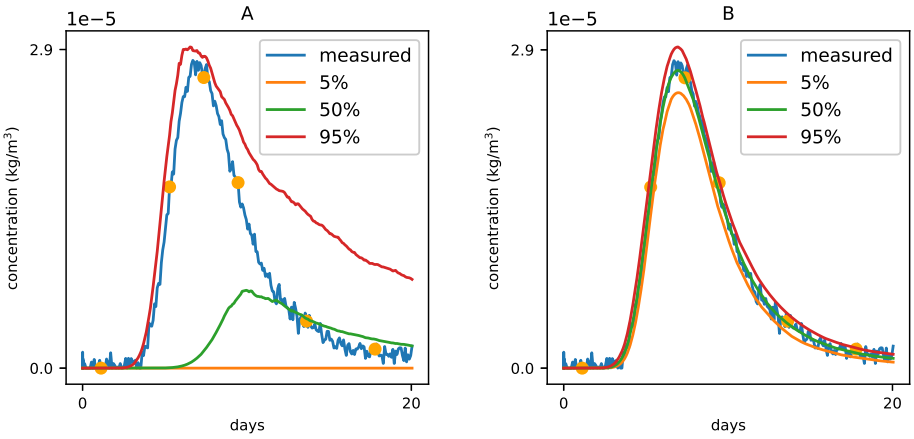

Figure 4.5: Prior (**A**) and posterior (**B**) tracer concentration quantiles. The orange dots correspond to data which were used for likelihood calculation. The posterior was obtained after 40 000 PoPEx iterations.

forward particle tracking module of flopy. The particle tracking was performed on the steady-state solution of the groundwater flow using the original grid composed of 100x100 pixels with the cell size of 5 m x 5 m. For each pixel in the domain, a probability of being in the 10 days zone was computed a priori (Figure 4.4A) and a posteriori (Figure 4.4B). Figure 4.5 shows the prior and posterior concentration quantiles at the pumping well.

Table 4.1: Physical properties associated with different geological facies

facies ordinal	1	2	3	4
facies	silt	fine sand	coarse sand	gravel
conductivity (m/s)	10^{-5}	10^{-4}	10^{-3}	10^{-1}
porosity	0.40	0.35	0.30	0.25
specific storage (m^{-1})	10^{-3}	5×10^{-4}	10^{-4}	10^{-5}

Application scenario

While it is possible to apply a classifier directly on a set of models $\mathbf{m}$, in the context of groundwater transport it is favorable to transform the input prior to feeding it to the ML scheme. Below we describe how ML inputs are formed and specify the algorithm parameters.

Transforming model into ML input

The groundwater velocity vector (v_x, v_y) controls advection and hydrodynamic dispersion

$$v_x = -\frac{K}{n_e}\frac{\partial h}{\partial x}, \quad v_y = -\frac{K}{n_e}\frac{\partial h}{\partial y} \tag{4.14}$$

with K the hydraulic conductivity, n_e the effective porosity and h the hydraulic head. The hydraulic heads are computed using a steady-state MODFLOW simulation. Values of conductivity and porosity are defined according to Table 4.1. Two-point numerical derivative is then used to calculate the hydraulic gradient.

Given the model $\mathbf{m}_i$ which is 100x100 pixels, X_i is formed by stacking v_x and v_y (each of size 98x98 pixels) and normalizing them to the interval [0,1], using one global minimum and maximum (the minimal/maximal value found in all vectors X_i, $i = 1 \dots n_t$). A 98x98 image with two channels is formed. The total number of features in the input is $98 \times 98 \times 2$. For AdaBoost and Random Forest the input is flattened, while CNN benefits from the spatial arrangement of the features. The classification labels are obtained by labeling fraction of t best likelihoods as good (1), and the rest not significant (0), as described in 4.2. The transformation only requires running a steady-state flow problem, whose computational cost is negligible when compared to the transport problem.

Initial PoPEx ensemble

We generated a PoPEx solution using only 500 iterations (Fig. 4.6). The initial ensemble is needed for training the machine learning scheme. The generated set of models with their likelihoods will be used to tune parameters of the machine learning schemes, and to select the best classifier for two different ratios $r = 0.4$ and $r = 0.2$. To this end, we will apply 5-fold cross-validation.

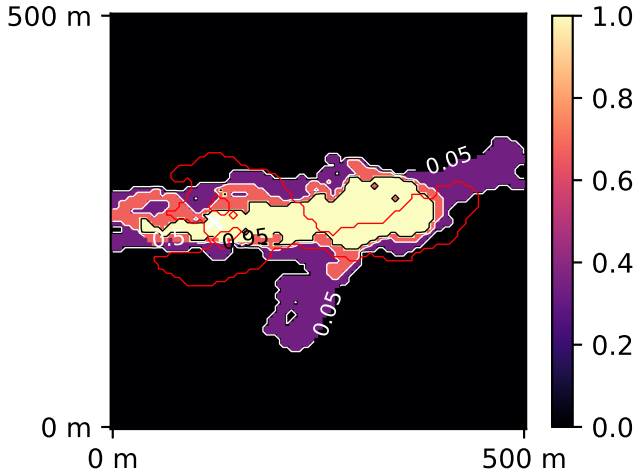

Figure 4.6: Posterior probability map of 10 days zone obtained after 500 PoPEx iterations. The red contour corresponds to the reference 10 days zone (simulated using the synthetic reality; considered unknown). The white cross represents the injection well.

For each ratio parameters of all methods will be re-tuned and the corresponding best classifiers will be used for validating the ML-accelerated PoPEx.

Test procedure

To test the methodology, we will compare the results obtained by the standard PoPEx algorithm after $N = 2\,000$ iterations with the results obtained by ML-accelerated PoPEx algorithm after $N = 4\,000$ iterations, taking into account computing time and error with respect to the reference solution. We will consider two cases: ratio $r = 0.2$ and $r = 0.4$. This test procedure will be repeated 4 times. Each time with a different initial PoPEx ensemble (neither being the ensemble used for ML parameter tuning).

To evaluate the results, the computing time and discrepancy with the reference solution (40 000 PoPEx iterations) will be computed. The discrepancy will be measured on the prediction: 10-days zone probability maps compared by means of the Jensen-Shannon (JS) divergence:

$$J(\widehat{\mu}||\mu^{ex}) = \frac{1}{2}\left(D(\widehat{\mu}||\bar{\mu}) + D(\mu^{ex}||\bar{\mu})\right) \tag{4.15}$$

with $\widehat{\mu}$ the estimated probability map, μ^{ex} the exact probability map, $\bar{\mu} = (\widehat{\mu} + \mu^{ex})/2$, and $D(\cdot||\cdot)$ Kullback-Leibler divergence, defined for the binary case as follows:

$$D(\mu_1||\mu_2) = \mu_1 \log \frac{\mu_1}{\mu_2} + (1-\mu_1) \log \frac{1-\mu_1}{1-\mu_2}. \tag{4.16}$$

Then, mean JS over the whole map summarizes the error with a unique value.

4.4 Results

In the first test, we perform a comparison of the performances of three machine learning algorithms: AdaBoost, Random Forest and CNN, and different proportions of models labeled as good (significant) models. This test corresponds to the practical application of the methodology to accelerate PoPEx: the choice of the ML method would be based on limited number of realizations without knowing the true solution. In the second test, we verify how the best performing methods according to the first step are able to accelerate the inversion.

Each inversion task was performed on a computing node composed of 2 Intel(R) Xeon(R) CPU D-1541 @ 2.10GHz CPUs running 64 threads in total. The total RAM of a node is 256 GB RAM. PoPEx was run with 63 processes (workers) and one master process. Running inversion with $N = 500$ iterations required about 24 h to finish. The approximate average time required for completing one geostatistical simulation by the computing node is $c_m = 6.3\,\mathrm{s}$ and for the forward solver: $c_g = 1.7 \times 10^2\,\mathrm{s}$; $c = \frac{c_m}{c_g}$ is approximately equal to 0.037.

Hyperparameter tuning

The hyperparameter tuning was performed with the initial ensemble containing 500 realizations. Depending on the ratio, different number of models were marked as good: 100 models for $r = 0.2$, 200 for $r = 0.4$. The validation scores from 5-fold cross-validation on these dataset were used to compare the models. The cases of two ratios are treated separately.

Random Forest and AdaBoost

The Random Forest and AdaBoost were implemented using scikit-learn library (Pedregosa et al. 2011). The mean 5-fold cross-validation *s-score* on the validation sets was used as performance metric. The Random Forest was tested with the following range of base estimators: 10 to 10 000. The number of samples (measured as fraction of all samples in the training dataset) were varied and scores reported for the ratio $r = 0.4$ (Fig. 4.7A) and $r = 0.2$ (Fig. 4.7C). The AdaBoost classifier was tested with the learning rates in the range 0.000 1 to 0.1 and number of estimators varied between 10 to 1 000. The scores for the ratios $r = 0.4$ and $r = 0.2$ are shown in Fig. 4.7B,D respectively. In the case $r = 0.4$ Random Forest slightly outperformed AdaBoost. The following parameters were selected: number of classifiers 1 000 and fraction of samples 0.2. In the case $r = 0.2$ AdaBoost outperformed Random Forest. The corresponding optimal parameters are: number of estimators 500 and learning rate 0.01. The performances of AdaBoost and Random Forest are similar, both methods suggest s-score around 2 for $r = 0.2$ and around 3 for $r = 0.4$.

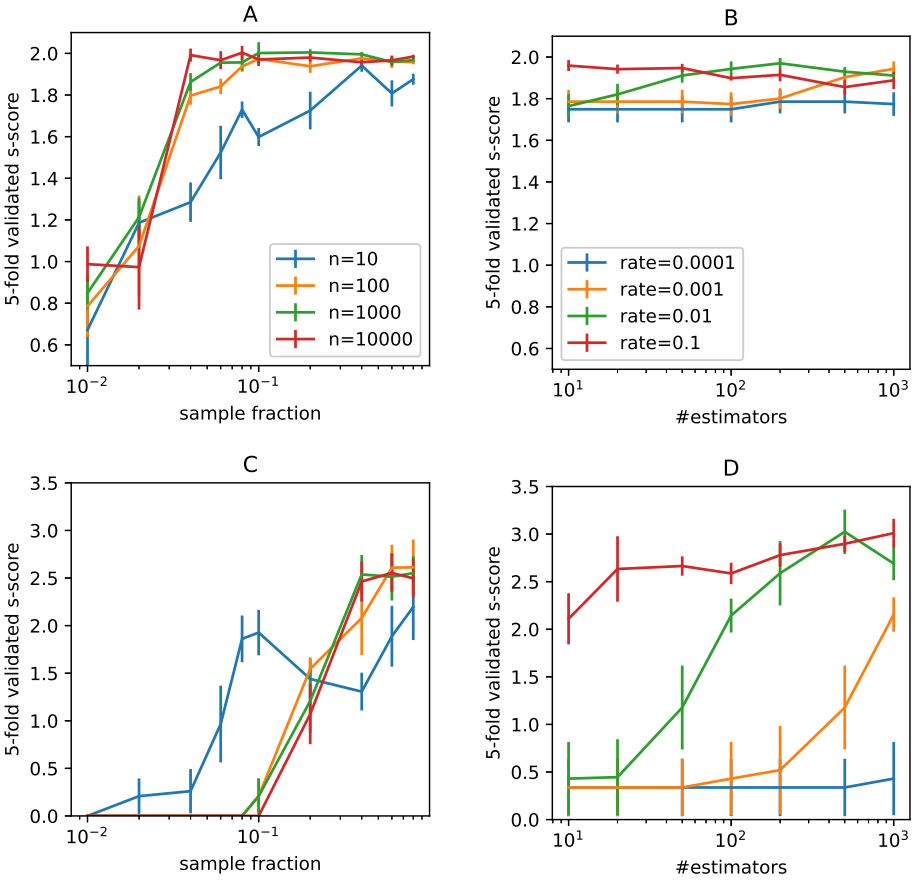

Figure 4.7: 5-fold cross-validated *s-scores* on test sets obtained by Random Forest (A,C) and AdaBoost (B,D) for two ratios 0.4 (upper row) and 0.2 (lower row). Mean values are reported with the standard mean error.

CNN architecture selection

The CNN was implemented using Keras library and the initial architecture (Tab. 4.2) is an adapted version of the AlexNet (Krizhevsky et al. 2012) to match the input size in our study. We considered the following variants of the initial architecture: included different blocks (1,2,3,4,5) and *kernel factors* (1,2,4,8). Five architecture variants were considered: first, including only block 5; second including block 4,5; third including blocks 3, 4, 5. As block 4 and 5 are dense layers, the architecture with one and two blocks correspond to feed-forward neural nets. The remaining architectures are CNNs with two dense layers before the output layer. We also introduced a second architectural parameter, *kernel factor*. It was used to divide the number of kernels of con-

Table 4.2: Layers of the convolutional neural network used for binary classification

block	layer, activation	kernel/pool size	out shape	#kernels or #nodes
input			(98,98,2)	
block1	conv. 2D, ReLU	5x5, stride 2x2	(47, 47, 24)	24
	max-pool	3x3, stride 2x2	(23, 23, 24)	
block2	conv 2D, ReLU	5x5, stride 2x2	(19, 19, 64)	64
	max-pool	3x3, stride 2x2	(9, 9, 64)	
block3	conv 2D, ReLU	5x5, stride 2x2	(7, 7, 96)	96
	conv 2D, ReLU	5x5, stride 2x2	(5, 5, 96)	96
	conv 2D, ReLU	5x5, stride 2x2	(3, 3, 64)	64
	max-pool	3x3	(1, 1, 64)	
block4	flatten			
	dense, ReLU		1024	1024
block5	flatten			
	dense, ReLU		1024	1024
output	dense, sigmoid		1	1

volutional layers and number of nodes in the dense layers. A higher number decreases the number of parameters of the neural net. In other words, the *kernel* factor defines a number by which the last column in the Tab. 4.2 is divided. In this way 20 variants of CNN architecture were formed. For each variant, a 5-fold cross-validated learning curve was obtained and the epoch corresponding to the minimal test loss was noted. The batch size was that of the full training set, and Adam optimizer was used with binary-cross-entropy loss and constant learning rate 0.001. The loss is weighted according to class weights: r for the class 0 (not significant models) and $1 - r$ for the significant ("good") models. The weights (as in the case of Random Forest) are used to correct the fact that the training set is not class-balanced; it forces the algorithm to pay more attention to the underrepresented class. Then, the 5-fold cross-validated *s-score* after the optimal epoch was retained. This procedure was repeated twice for ratio $r = 0.4$ and $r = 0.2$ (Fig. 4.8A, B) respectively. Simpler architectures perform generally better than those containing all the blocks and simple feed-forward nets outperform the more complex architectures.

Choosing ML algorithm

The 5-fold cross-validated scores of tuned (with selected optimal hyperparameters, as in Tab. 4.3) AdaBoost, Random Forest and CNN algorithms are reported in Tab. 4.4. The following mean scores and standard errors of the mean are compared: *s-score* (according to the formula 4.10), precision, recall and approximate *s-score* (precision/r). The *s-score* value, as explained in the

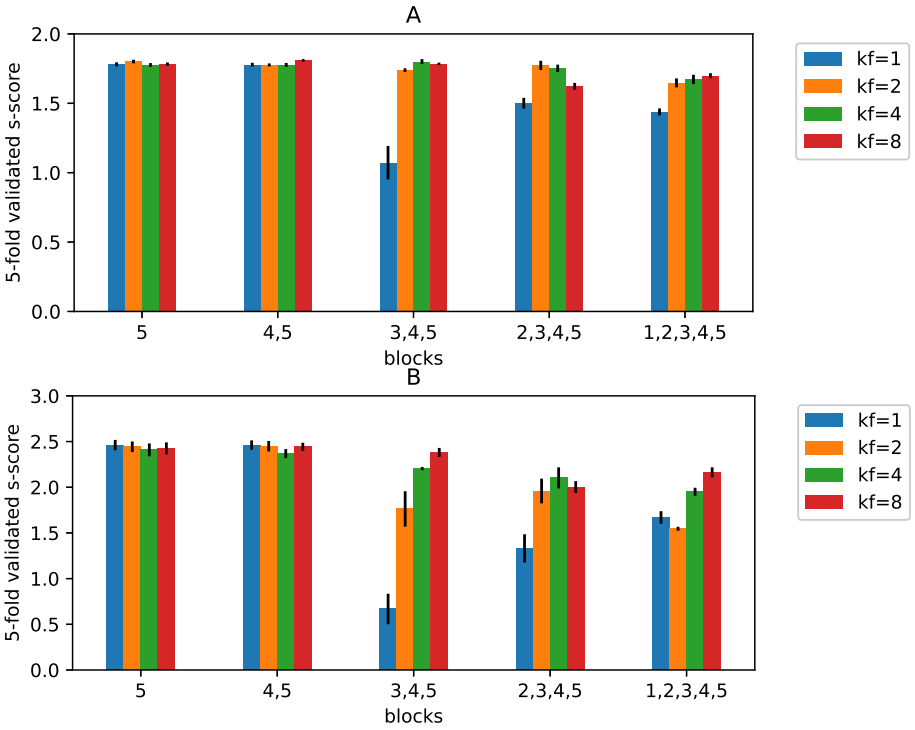

Figure 4.8: 5-fold cross-validated *s-scores* on test sets obtained by CNN for two ratios 0.4 (A) and 0.2 (B). Mean values are reported with the standard mean error.

Table 4.3: Selected optimal hyperparameters for each ML method and each ratio

classifier	r	hyperparameters
AdaBoost	0.4	#estimators=200, rate=0.01
RF	0.4	#classifiers=1000, fraction=0.2
CNN	0.4	blocks=4,5, kf=8
AdaBoost	0.2	#estimators=500, rate=0.01
RF	0.2	#classifiers=1000, fraction=0.4
CNN	0.2	blocks=5, kf=1

methodology, can be interpreted as an optimistic estimator for the real speed-up value.

Table 4.4 shows that the *s-score* and the precision obtained with Random Forest and AdaBoost are higher than those obtained with CNN for both values of r. The recall values are similar for all classifiers for $r = 0.4$, and the recall for

Table 4.4: 5-fold cross-validated performance of classifiers with the optimal selected hyperparameters on the 500 test set

classifier	r	*s-score*	precision	appr. *s-score*	recall
AdaBoost	0.4	1.98 ± 0.03	0.84 ± 0.02	2.10 ± 0.03	0.78 ± 0.05
RF	0.4	2.01 ± 0.02	0.86 ± 0.02	2.15 ± 0.03	0.77 ± 0.03
CNN	0.4	1.82 ± 0.01	0.76 ± 0.01	1.91 ± 0.03	0.80 ± 0.03
AdaBoost	0.2	3.1 ± 0.3	0.77 ± 0.06	3.9 ± 0.3	0.48 ± 0.06
RF	0.2	2.6 ± 0.2	0.95 ± 0.05	4.8 ± 0.3	0.21 ± 0.04
CNN	0.2	2.48 ± 0.06	0.56 ± 0.03	2.8 ± 0.2	0.63 ± 0.05

$r = 0.2$ is better for CNN than for AdaBoost and Random Forest. It means that AdaBoost and Random Forest were able to achieve a high proportion of good models among all labeled as good, but missed some good models. Consequently, these results confirm that the approximate *s-score* overestimated the actual ones.

Test of ML-accelerated PoPEx

Random Forest with 1000 estimators and samples fraction 0.4 was chosen for validating ML-accelerated PoPEx for ratio of 0.4, and AdaBoost with 500 estimators and learning rate 0.001 for validating ML-accelerated PoPEx for ratio of 0.2. In the validation experiment, PoPEx in the standard mode was run for $N = 2\,000$ iterations and two ML-accelerated PoPEx versions were run for $N = 4\,000$ iterations: RF-accelerated PoPEx with $r = 0.4$ and AdaBoost-accelerated PoPEx with $r = 0.2$.

For each of these three PoPEx modes, 4 independent runs were performed and averages used to compute the mean errors with respect to the reference solution. The total running times, and the number of models fed to the forward solver were recorded. The accelerated modes of PoPEx were trained once after $n_t = 500$ iterations and from that iteration, the ML schemes were used.

Example results (1 out of 4) are shown in Fig. 4.9, where predictions of 10-days zones after all iterations (2 000 for the standard mode, 4 000 for ML accelerated) are reported with Jensen-Shannon error maps. The reference solution obtained after 40 000 iterations (Fig. 4.4) was used in the Jensen-Shannon formula (4.15). In this example, the exact solution took 94h 45 min to compute, the RF-accelerated with $r = 0.4$ took 88 h 12 min, and the AdaBoost-accelerated with $r = 0.2$ only 47 h 40 min. The lowest mean JS error achieved RF-accelerated mode with $r = 0.4$: 0.010, the exact mode: 0.015 and AdaBoost-accelerated mode with $r = 0.2$: 0.014.

Figure 4.10 shows the convergence of the different PoPEx modes with respect to the number of iterations. As expected, ML schemes slow down the convergence

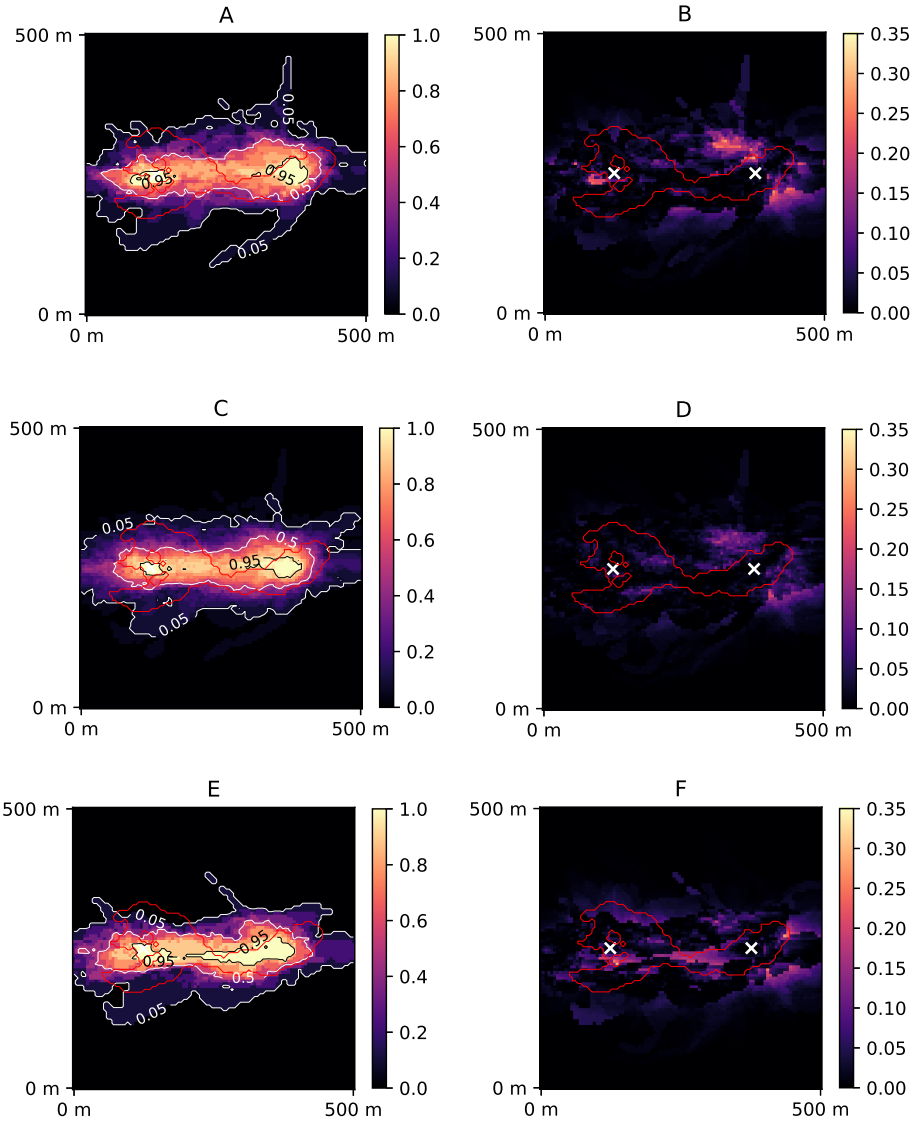

Figure 4.9: Posterior probability maps of 10 days zone with Jensen-Shannon error maps for standard PoPEx solution with $N = 2\,000$ (**A, B**), for RF-accelerated PoPEX with $N = 4\,000$ and $r = 0.4$ (**C, D**), and for RF-accelerated PoPEx with $N = 4\,000$ and $r = 0.2$ (**E, F**). The red contour corresponds to the reference 10 days zone.

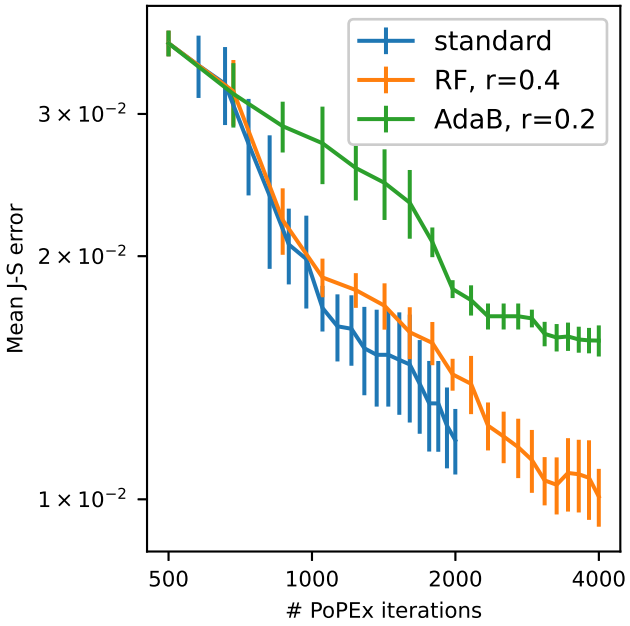

Figure 4.10: Convergence of standard PoPEx method compared with RF-accelerated PoPEx with ratio $r = 0.4$ and AdaBoost-accelerated with $r = 0.2$. Each point is an average over 4 PoPEx runs, and error bars show standard deviation.

due to incorrect model classifications. Nevertheless, the convergence rate of the RF-accelerated mode with $r = 0.4$ is close to the standard mode. We can also see that this accelerated mode achieves the lowest values of mean error. Not all the generated models in ML-accelerated modes are sent to the forward solver, therefore the time required to achieve a specified number of iterations vary. This is why it is more representative of computing times to plot convergence with respect to the total number of models fed to the forward solver (Fig. 4.11A) or to the total computing time (Fig. 4.11B). These plots show that both ML-accelerated methods need computing fewer forward models to achieve the same error as the standard PoPEx version (Fig. 4.11). These graphs show also that the convergence rate is similar for both values of r. If converted to total computing time, the AdaBoost-accelerated mode with $r = 0.2$ and RF-accelerated mode with $r = 0.4$ require systematically less computing time than the standard mode.

Figure 4.12 presents speed-up for different mean J-S errors. The speed-up was calculated as follows. It is the ratio of the computing time of the standard PoPEx algorithm by the computing time of ML-accelerated PoPEx for obtaining the same mean J-S errors. The simulation time was counted from iteration 500 to the iteration by which a specified mean J-S error was achieved. ML-

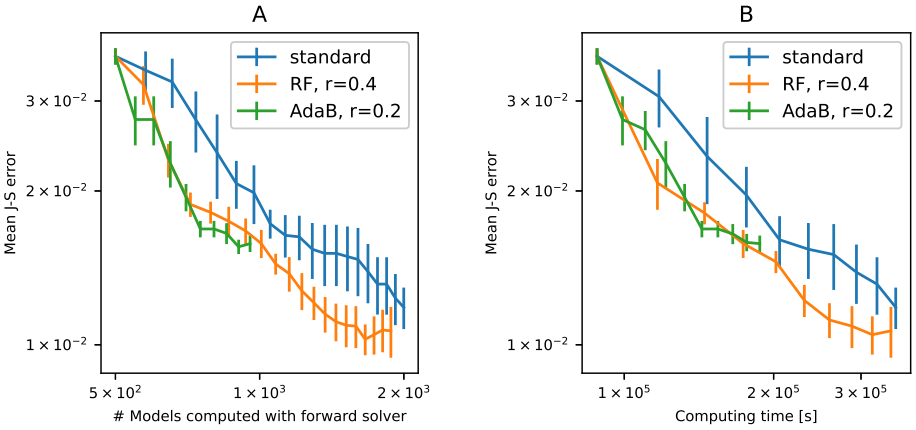

Figure 4.11: Error with respect to the number of models computed by the forward solver for standard PoPEx, RF-accelerated PoPEx with $t = 0.4$, and AdaBoost-accelerated with $r = 0.2$. Each point is an average over 4 PoPEx runs, and error bars show standard deviation.

accelerated modes achieve peak speed-ups greater than 2x for large errors, that is at the beginning of inversion procedure. The speed-ups for both ratios are similar and the *s-score* estimates well the real speed-up rate for $r = 0.4$. For $r = 0.2$ the *s-score* overestimated the speed-up.

4.5 Discussion and conclusion

In this article, we introduced a generic framework for accelerating the posterior population expansion algorithm (PoPEx) using machine learning techniques, and tested it on a groundwater transport problem. Our approach is generic and does not interfere with the geostatistical method used for modeling the prior and can be adapted to any type of forward problem. We found that in our set up, both Random Forest and AdaBoost performed well for classifying if generated models are of significance. While AdaBoost and Random Forest performed similarly for the two ratios, we recommend labeling a relatively high ratio of models as significant based on their likelihood — the training suffers from lower variations and results in more stable models. The obtained speedups are close to 2 for this problem, which can represent a significant gain of computing time and resources for large problems.

Random Forest and AdaBoost performance were comparable and superior to CNN in the classification task. Kelleher et al. (2015) pointed out that Random Forests perform well when the input has many features, as is the case in our study; AdaBoost performed well for the more imbalanced case ($r = 0.2$). The poor performance of CNN is surprising, as it benefits from information about

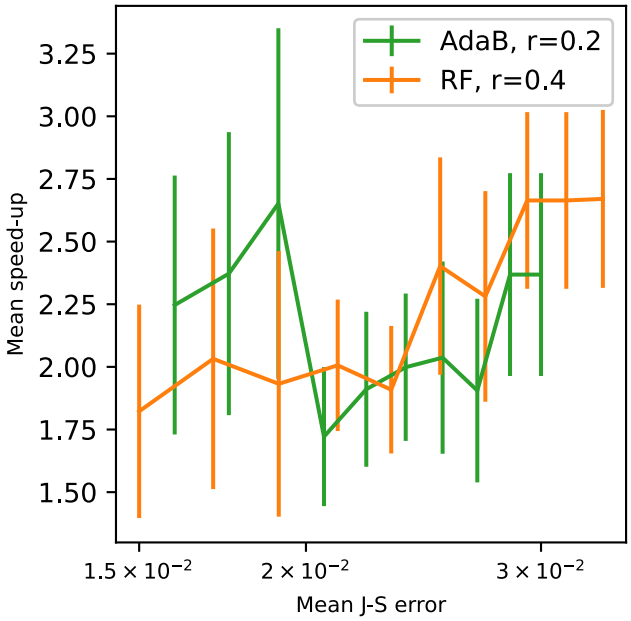

Figure 4.12: Speed-up with respect to the mean Jensen-Shannon error standard PoPEx: for RF-accelerated PoPEx with $r = 0.4$, and AdaBoost-accelerated with $r = 0.2$. Each point is an average over 4 PoPEx runs, and error bars show standard deviation.

the spatial arrangement of the input data. However, while CNN is very effective in recognizing shapes which are invariant by translation or rotation, in our specific case the absolute position of the features is important to predict the solute transport processes. The fact that simplified CNN architectures (which were reduced to feed-forward neural nets) performed generally better than the more complicated architectures, supports this claim. Further research is needed to determine if it is possible to design a specific Neural Network architecture outperforming the methods presented in this paper. Another important aspect is that the training dataset including only 500 elements is probably too small for the neural network to achieve good performance.

Another challenge for the ML methods is that the labels are arbitrary and depend on the relative values of the likelihood. The generated models are not "good" or "bad" in an absolute sense; they are "better" or "worse" than the others. It is possible that the same model receives different labels depending on other models in the ensemble. The imbalanced nature of the dataset makes it harder to train, especially for low values of the ratio r. We therefore recommend using the ratio $r = 0.4$, as it makes training easier and the performance of the classifier is more robust. Lower ratios are attractive not only from the point of view of the estimated speed-up, but also because lower ratios correspond to

fewer good models, which is more relevant in terms of likelihood statistics (few models represent the majority of posterior weights). However, lower ratios suffer from the discrepancy between the estimated speed-up and the actual speed-up. This observation can be explained by the fact that rejecting a good model slows down the convergence of PoPEx. If a classifier miss-classifies a good model, the core PoPEx algorithm misses an important information which could guide the sampling. PoPEx then continues sampling from a less-informed distribution, proposing worse models. This is why using a classifier with a low recall (low ratio value) is more penalizing than when using one with a higher recall (higher ratio values).

The proposed classification method could be employed in other Monte Carlo schemes, such as rejection sampling, importance sampling, or Metropolis algorithm (Tarantola 2005). It could even yield better results than with PoPEx algorithm, as PoPEx is iteratively expanding the ensemble of models by learning from the already sampled models. Simpler samplers without memory of previously generated models could benefit more from the learning schemes.

Finally, an interesting alternative to using classifiers could be to use regression techniques to directly estimate the likelihood value. However, this would be a challenging task, as wrong likelihood estimations could mislead the sampler. It could even lead to slowing it down if irrelevant models would receive erroneously high likelihood. In the present work, erroneous classifications only slow down the inversion but they are not a source of noise.

The datasets generated for this study can be found in the github repository: `https://github.com/randlab/ml-popex`.

Chapter 5

Tempered likelihood for Posterior Population Expansion

5.1 Introduction

In the previous chapter, we studied a synthetic tracer test problem and saw how the posterior population expansion (PoPEx) algorithm was used to solve this inverse problem. Let us note that in the inversion process, we used only a subset of the synthetic observation data. The original data was a time series of tracer concentration containing 276 points, which was corrupted with white noise of a constant variance. However, the likelihood formula used in that inverse problem (Sec. 4.3) was written as proportional to:

$$\exp\left(-\frac{1}{2\sigma_{\text{obs}}^2}\sum_{i\in\mathcal{I}_s}\left(\mathbf{g}(\mathbf{m})[i]-\mathbf{d}_{\text{obs}}[i]\right)^2\right), \tag{5.1}$$

with $\mathbf{d}_{\text{obs}} \in \mathbb{R}^{276}$ the observed data, $\mathbf{g}(\mathbf{m}) \in \mathbb{R}^{276}$ the simulated data using model $\mathbf{m} \in \mathfrak{M}$, $\mathfrak{M}$ the model space. and $\mathcal{I}_s$ the subset of indexes defined as: $\mathcal{I}_s = \{50, 100, 125, 150, 200, 250\} \subset \mathcal{I}$, and σ_{obs}^2 variance of the measurement error. We used square brackets to indicate the elements of vector, for example $\mathbf{d}_{\text{obs}}[5] \in \mathbb{R}$ is the fifth element (scalar value) of the vector $\mathbf{d}_{\text{obs}}$. The difference between this *sampled* likelihood formula and the correct likelihood is that the subset $\mathcal{I}_s$ (containing only 6 data points) was used instead of all data indexes $\mathcal{I} = \{1, 2, \ldots, 276\}$. It corresponds to leaving some data out and performing inversion only on the available subset.

The same approach for dealing with concentration data was applied in one of the original PoPEx papers. The motivation for using this *sampled* likelihood formula is that when the correct and complete expression is used, the likelihood values are zero for most realizations because of numerical approximations. It becomes intractable to compare their quality based on these values. It results in a poor representation of uncertainty and a slow convergence of the

method (Jäggli et al. 2018). Laloy et al. (2018) also pointed to the problem of *peakedness* of the likelihood function, meaning that the function has a sharp peak, and therefore is hard to explore using Monte Carlo techniques.

While in the case of a simple unimodal breakthrough curve, leaving out some data could be an acceptable approach to reduce the dimensionality of the data, it makes less sense in more complex situations such as for spatially distributed measurements piezometric data or more complex time series. Choosing a subset of observations would be cumbersome and potentially bias the inversion results. Another shortcoming of this technique is the fact that it is combined with another approximation, correction of the predictive weights, as described in the section 4.2. In PoPEx framework, the adaptive importance sampling (AIS) weights need to be corrected following equation (4.4), see page 66. Despite using this technique of correcting the weights, the sampled likelihood must have been used. While the technique of *corrected* weights converge with the number of iterations to the exact AIS weights, the sampled likelihood formula does not change and thus excludes permanently some data from inversion.

Therefore, a more generic approach would be useful for approaching inversion cases with more data points, and we propose to use a tempering term in the likelihood function for PoPEx predictions and include all the data in the inversion. The tempering term inflates artificially the variance of the noise in the data and is adapted to match a desired number of effective weights in PoPEx. It has two advantages: first, if there is a solution of the inverse problem, where sufficiently many significant likelihoods are obtained, the factor would tend to 1, and the correct likelihood will be used. Second, no longer *correction* of the weights is needed.

Our approach has been inspired by other solutions of the problem of the *peakedness* of the likelihood function. Laloy et al. (2018) presented a case of 3-D Transient Hydraulic Tomography, where 1568 data points were used in the inversion using DREAM-ZS algorithm (Laloy and Vrugt 2012). The inversion was stopped before reaching convergence criterion. In that study, tempering of the likelihood function was implemented, but limited to the burn-in. It consisted in using an inflated variance term in the likelihood function. A similar technique to tempering is also used in the context of data assimilation. Lam et al. (2020) used ensemble smoother with multiple data assimilation (ESMDA) for discrete inversion, where geostatistical simulation uses pyramids. ESMDA applies repetitively Kalman update to assimilate data, but introduces factor α for reducing the correction term, as the same data is assimilated multiple times (Emerick and Reynolds 2012). It corresponds to reducing the confidence given to the (noisy) data at every iteration of the data assimilation.

5.2 Posterior event prediction

The posterior measure function $\sigma(\mathbf{m})$ serves to compute predictions, in other words the quantities of interest or posterior measures of events. For example, it can be a probability map of facies, or a probability map of 10-day groundwater protection zone, as seen in the previous chapter. The expectation value μ of any measurable quantity of interest $f(\mathbf{m})$ can be expressed using the following integral (as we have already seen, 4.2):

$$\mu = \int_{\mathfrak{M}} \sigma(\mathbf{m}) f(\mathbf{m}) d\mathbf{m}.$$

The adaptive importance sampling provides a convenient formula, the self normalized estimator $\hat{\mu}_{\text{sn}}$, to approximate integrals of this type using the following sum (Jäggli et al. 2018):

$$\hat{\mu}_{\text{sn}} = \sum_{j=1}^{k} f(\mathbf{m}^j)\tilde{w}^j. \tag{5.2}$$

The $\tilde{w}^j$ are normalized weights: $\tilde{w}^j = w^j / \sum_{i=1}^{k} w^j$, with k representing the total number of generated models (iterations). The superscript is not used as power, instead it is used for indexing iterations, we keep this notation for consistency with the reference PoPEx paper (Jäggli et al. 2018). In the AIS framework, the weights w^k are given by:

$$w^k = \frac{\sigma(\mathbf{m}^k)}{\phi^k(\mathbf{m}^k)} = c\frac{\rho(\mathbf{m}^k)}{\phi^k(\mathbf{m}^k)} L(\mathbf{m}^k), \tag{5.3}$$

with a constant c (that can be ignored later due to self-normalization), $L(\mathbf{m})$ the likelihood function, and $\rho(\mathbf{m})$ the prior measure. ϕ^k is a sampling distribution which is updated at every iteration k. The main idea of adaptive importance sampling is to update ϕ^k in a way that it resembles σ but has heavier tails.

5.3 Tempering for posterior event prediction

To resolve the problem of few significant weights, we suggest an approach based on *tempered* likelihood function. It is similar to using higher error variance in the likelihood formula. The tempering factor is adapted (optimized) based on the desired number of significant models.

Tempered likelihood

Let us define the family of *tempered* likelihood function $L_{\text{t}}(\mathbf{m}; f_\sigma)$:

$$L_{\text{t}}(\mathbf{m}; f_\sigma) = \exp\left[\frac{1}{f_\sigma^2} \log\left(L(\mathbf{m})\right)\right]. \tag{5.4}$$

with the tempering factor $f_\sigma \geq 1$. If $f_\sigma = 1$ we have: $L_{\mathrm{t}}(\mathbf{m}^i; 1) \equiv L(\mathbf{m}^i)$, and the tempered likelihood becomes equivalent to the standard likelihood function. In this sense, the tempered likelihood is a generalization of the correct likelihood function for the problem at hand. The tempering factor reduces confidence in the data, it can be interpreted as a factor inflating the variance of the measurement error.

If we use the soft likelihood in (5.3), we obtain a parametric formula for tempered weights:

$$w_{\mathrm{t}}^k(f_\sigma) = c\frac{\rho(\mathbf{m}^k)}{\phi^k(\mathbf{m}^k)} L_{\mathrm{t}}(\mathbf{m}^k; f_\sigma). \tag{5.5}$$

Finally, it leads to the tempered formula for the self-normalized estimator:

$$\hat{\mu}_{\mathrm{sn,t}}(f_\sigma) = \sum_{j=1}^{k} f(\mathbf{m}^j)\tilde{w}_{\mathrm{t}}^j(f_\sigma). \tag{5.6}$$

Optimal tempering factor

The tempering factor can be chosen arbitrarily, for example $f_\sigma = \sqrt{N}$ would correspond to taking the average of log-likelihood over N observation points of the mismatch. Instead of fixing a value for f_σ, we propose a method to adaptively choose optimal f_σ. It is inspired by the formulation of *corrected* PoPEx weights.

Let us consider a set $\mathcal{W}^k(f_\sigma)$ of k tempered weights with parameter f_σ:

$$\mathcal{W}^k(f_\sigma) = \left\{w_{\mathrm{t}}^1(f_\sigma), \ldots, w_{\mathrm{t}}^k(f_\sigma)\right\}. \tag{5.7}$$

The effective sample size, already introduced in (4.6) for the set $\mathcal{W}^k(f_\sigma)$ is given by:

$$n_e(\mathcal{W}^k(f_\sigma)) = \frac{(\sum_{i=1}^k w_{\mathrm{t}}^i(f_\sigma))^2}{\sum_{i=1}^k (w_{\mathrm{t}}^i(f_\sigma))^2}. \tag{5.8}$$

Suppose that the target value of the minimal number of effective weights θ is chosen by the user, who also specifies the value of $f_{\max}$ which will be the max bound for f_σ. We will define f_σ as optimal if it is such that number of effective weights n_e equals at least θ and $f_\sigma \in [1, f_{\max}]$ is as small as possible. This can be translated into the following optimization problem:

$$f_{\mathrm{opt}} = \underset{f_\sigma \in [1, f_{\max}]}{\arg\min} \left(n_e(\mathcal{W}^k(f_\sigma)) - \theta\right)^2, \tag{5.9}$$

where f_{opt} is the optimal tempering factor.

Optimal tempered weights

The set of the optimal tempered weights is given by $\mathcal{W}^k(f_{\text{opt}})$, and after normalization they can be used in (5.6) to get the desired estimator.

The tempering framework can be summarized in the form of an algorithm. It takes as input: θ — the target number of effective weights; $f_{\max}$ — the max bound for the tempering factor. The algorithm is as follows:

OPTIMAL-TEMPERED-WEIGHTS($\theta, f_{\max}$)

1 Minimize $\left(n_e(\mathcal{W}^k(f_\sigma)) - \theta\right)^2$ subject to $f_\sigma \in [1, f_{\max}]$
2 f_{opt} = argument of the minimum
3 Compute $\mathcal{W}^k(f_{\text{opt}})$
4 **for** $j = 1, \ldots, k$
5 $\quad \tilde{w}_{\text{t}}^j = w_{\text{t}}^j / \sum_{i=1}^k w_{\text{t}}^i$
6 **return** $\{\tilde{w}_{\text{t}}^1, \ldots, \tilde{w}_{\text{t}}^k\}$

5.4 Results of the test case

To test the tempering technique, we used the case introduced in Section 4.3. The aim is to predict the 10 days groundwater protection zone after having inverted a tracer test. PoPEx was run for 4 000 iterations for every of three approaches that we compare.

The first, baseline, approach corresponds to using the exact likelihood and corrected weights with $l_0 = 10$ (Fig. 5.1A) and with $l_0 = 100$ (Fig. 5.1B). l_0 is the number of effective weights and its meaning is the same as θ, but we use this symbol to indicate that this parameter is linked to *corrected* weights (as opposed to *tempered* weights for which we use θ). Figs. 5.1A and B illustrate the motivation of using an alternative approximation technique, as uncertainty is very poorly represented.

The second approach (as described in Section 4.3) with *sampled* likelihood and *corrected* weights improves the uncertainty representation, both with $l_0 = 10$ (Fig. 5.1C) and with $l_0 = 100$ (Fig. 5.1D). The third approach is inversion using all data points and *tempered* weights with $\theta = 10$ (Fig. 5.1E) and $\theta = 100$ (Fig. 5.1F).

In all three approaches, the prior is the same. The first and the third approach are based on the same sampling mechanism, as no tempering of the likelihood is used during the inversion process. The tempering only happens when generating predictions. Fig. 5.2 shows the posterior matches of the concentration curves. The posterior concentration curves practically reduce to a single one for the exact likelihood and corrected weights with $l_0 = 10$ (Fig. 5.2A) and with $l_0 = 100$ (Fig. 5.2B). Sampled likelihood and corrected weights give a median

curve matching the data and wider confidence intervals with $l_0 = 10$ (Fig. 5.2C) and with $l_0 = 100$ (Fig. 5.2D). Tempered likelihood with $\theta = 10$ gives similar data match and confidence interval (Fig. 5.2E), but with $\theta = 100$ the uncertainty is overestimated and median curve does not match the observations very well (Fig. 5.2F).

For the *tempered* likelihood, we checked how the optimal tempering factor changes with the number of PoPEx iterations (Fig. 5.3). The value of $f_{\max}$ was fixed to 1 000.

5.5 Discussion

Both *sampled* likelihood with *corrected* weights and *tempered* weights (with all data points) can produce reasonable prediction of protection zone (Fig. 5.1). From Fig. 5.1A,B it is clear that using only *corrected* weights is not sufficient, as the number of significant models is severely reduced and cannot be improved. In the case of sampled likelihood with *corrected* weights, increasing l_0 from 10 to 100, did not have much effect (Fig. 5.1C,D). The tempering framework gives more flexibility, as increasing θ has a clear effect on the predicted zone (Fig. 5.1F), as compared with Fig. 5.1E.

Combining large θ and $f_{\max}$ allows giving significantly more confidence in prior, which might be undesirable. If we compare Fig. 5.2E and Fig. 5.2F, we can notice that the latter has inferior data match, with median not very close to the measured data. We did not notice such a departure from median in Figs. 5.2C and D. Therefore, when applying the *tempered* weights, θ parameter should not be chosen too large. $\theta = 10$ seems to be a good default parameter. A strategy to overcome the overconfidence in prior is also setting $f_{\max}$ to a smaller value. Finally, the optimal tempering factor can be interpreted as a rough convergence indicator. Setting a large θ can slow down convergence, as suggests Fig. 5.3, where $\theta = 100$ results in higher f_σ values than with $\theta = 10$. It is coherent with the observation that the data match with $\theta = 100$ was not very good (Fig. 5.2F) and the protection zone less symmetric, and more relying on prior information (Fig. 5.1F) — compare with Fig. 4.4.

5.6 Conclusion

In this chapter, we introduced tempered likelihood, which solves the problem of *peakedness* of the likelihood function and allows having a better uncertainty representation in predictions for PoPEx algorithm. It consists essentially in performing a postprocessing step after running the inversion with all the available data. The framework is similar to that of *correcting* PoPEx weights (Jäggli et al. 2018), but its main advantage is the fact that all the data can be used for inversion, regardless of the size of the observation set. An additional benefit is that it offers more flexibility, since it can give even more confidence to prior.

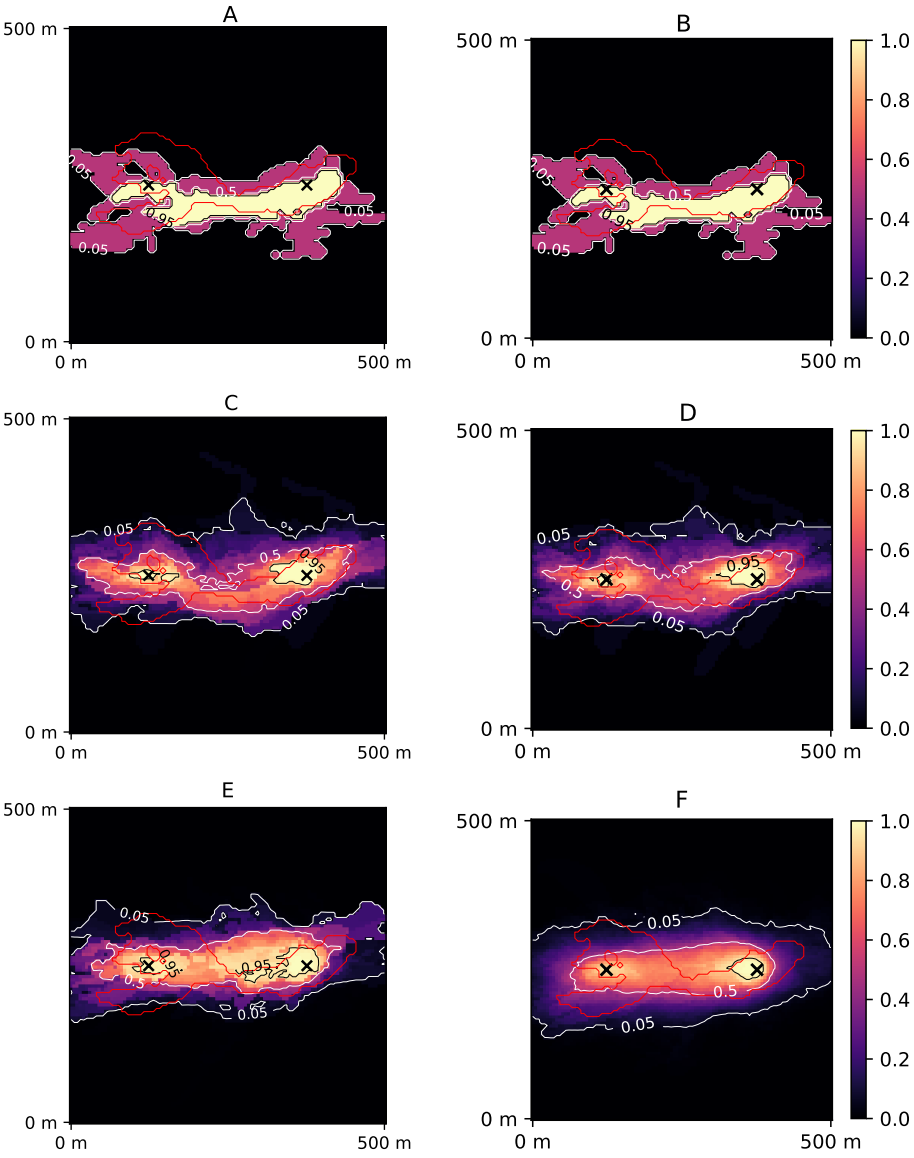

Figure 5.1: Predictions of 10 day groundwater protection zone. The upper row corresponds to exact likelihood and corrected weights with $l_0 = 10$ (A), and $l_0 = 100$ (B); the middle row to sampled likelihood with corrected weights with $l_0 = 10$ (C), and $l_0 = 100$ (D); and the bottom row to tempered weights with $\theta = 10$ (E), and $\theta = 100$ (F).

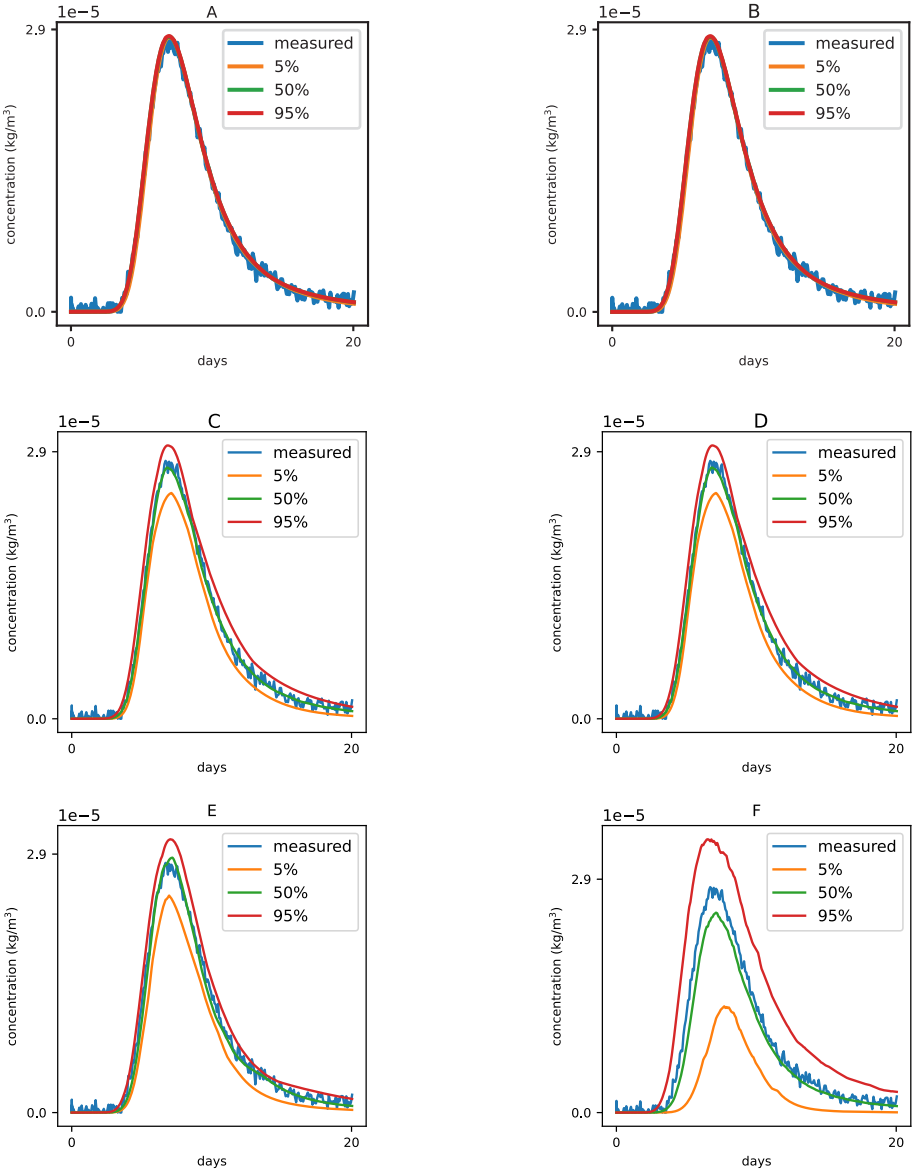

Figure 5.2: Posterior data match of the concentration series. The upper row corresponds to exact likelihood and corrected weights with $l_0 = 10$ (A), and $l_0 = 100$ (B); the middle row to sampled likelihood with corrected weights with $l_0 = 10$ (C), and $l_0 = 100$ (D); and the bottom row to tempered weights with $\theta = 10$ (E), and $\theta = 100$ (F).

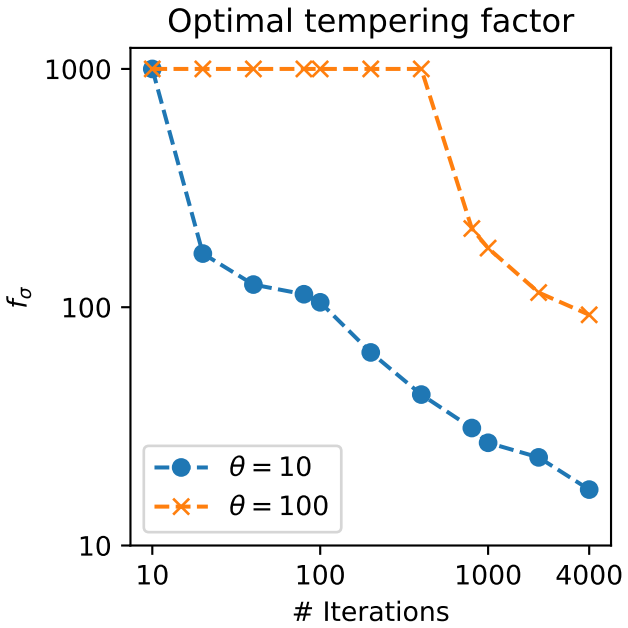

Figure 5.3: Optimal tempering factors obtained with $f_{\max} = 1000$ and two θ values with respect to the number of PoPEx iterations. The inversion was performed with all the 276 data points.

The automated algorithm for finding the tempered weights requires only two parameters: the desired number of effective weights, and the maximal value for the tempering factor.

The use of the tempered likelihood is not restricted to importance sampling techniques. It can be used also in other Monte Carlo inversion frameworks, such as DREAM-ZS (Laloy et al. 2018). and fixed tempering factors should be chosen instead.

Chapter 6

Comparison of three recent discrete stochastic inversion algorithms on a pumping test case

6.1 Introduction

Solving inverse problem has become an indispensable task for groundwater modelling. In Chapter 4, we have seen an example of a hydrogeological inverse problem, where tracer concentration data was used to reconstruct the possible subsurface properties. We have treated that discrete problem in a stochastic manner to be able to quantify uncertainty of the prediction, and generate a probabilistic prediction of 10 days groundwater protection zone. The uncertainty quantification is desirable for reasonable groundwater management and risk assessment, well-informed decisions, and better aquifer management.

To constrain the stochastic inversion with geological knowledge often requires using a geostatistical and discrete model of geology (Linde et al. 2015). Among the state-of-the-art geostatistical methods allowing to represent categorical fields, the multiple-point statistics (MPS) approach is based on training images (Mariethoz and Caers 2015). Its advantage is that it respects high order statistics and allows a flexible control of heterogeneities. MPS algorithms have been extensively used in inversion frameworks. Early examples include for example the probability perturbation method (Caers and Hoffman 2006), the blocking moving window algorithm (Alcolea and Renard 2010; Hansen et al. 2012), or the iterative spatial resampling (Mariethoz et al. 2010c). These methods iteratively update MPS realizations by imposing hard or soft conditioning data either in an optimization or Monte Carlo Markov chains perspective. A more recent development is the Posterior Population Expansion (PoPEx) algorithm. It is an adaptive importance sampling (AIS) scheme that also uses hard conditioning data to iteratively expand an ensemble of models (Jäggli et al. 2017; Jäggli et al. 2018). An important feature of PoPEx is that it is highly

parallelizable.

An emerging alternative to MPS are generative adversarial nets (GAN), which gained in popularity in recent years due to their ability to generate images of any types provided sufficiently large training dataset and relatively low dimensional latent space representation. The representation of parameters in the latent space (which is often gaussian) is convenient for Monte Carlo algorithms. In the hydrogeological context, one of the most notable examples is DREAM-ZS algorithm combined with spatial GAN (Laloy et al. 2018). In a broader geophysical context, GANs were used for stochastic seismic inversion (Mosser et al. 2019).

Another important emerging methodology is data assimilation. It has a different theoretical framework, as stems from multi-Gaussian fields and was later applied to non-Gaussian and categorical examples (Kang et al. 2019; Oliver and Chen 2018; Zhou et al. 2014). Recently, Lam et al. (2020) applied ensemble smoother with multiple data assimilation (ESMDA) to condition Gaussian pyramids in the multi-resolution Direct Sampling MPS simulations (Straubhaar et al. 2020). Their approach enabled data assimilation with categorical fields and generate ensemble to quantify uncertainty.

More generally, the current research in this field has seen the development of many novel inversion techniques to integrate complex prior distributions of discrete variables, but it is not always clear what are the advantages and limitations of the different methods. Previous inter comparison exercises (Hendricks Franssen et al. 2009; Zimmerman et al. 1998) did not consider the case of discrete fields with geological prior knowledge. One aim of this chapter is therefore to provide such a comparison for the 3 methods that we described in the previous paragraphs. For the comparison to be fair, we re-implemented the three methods using similar tools. Once again, the code is available in github.

When comparing the methods, it is important to consider the impact of the choice of the prior distribution, as in a real data scenario, the correct prior is not known. The inversion must be performed with a geological prior which might not represent fully the real geology. Therefore, the question of importance of the prior in the stochastic inversion should be addressed to know how the different methods are able to cope with imperfect priors.

To answer these questions, we introduce a synthetic pumping test experiment, and we apply three different algorithms: PoPEx, DREAM-ZS and ESMDA to compare their performance with respect to the reference solution. We also test each method with different priors: the correct prior and two priors diverging from the original one. The priors are defined by different training images and geostatistical algorithm parameters. The training images are different for each case, but the simulation parameters are fixed.

6.2 Inversion algorithms

Three stochastic inversion algorithms are compared in this chapter. In this section, we provide a detailed description of the algorithms, their implementations and the tools used to generate the discrete random fields.

Stochastic formulation

In section 4.2, we already introduced the stochastic formulation of the inverse problem and the PoPEx method. Here, we recall the main notations because we will use them to present the three algorithms.

The observed data are stored in a vector of real values $\mathbf{d}^{\text{obs}} \in \mathbb{R}^N$, and N is the number of observed data points. Let us consider a model manifold $\mathfrak{M}$. Any model $\mathbf{m} \in \mathfrak{M}$ is supposed to describe fully the physical system. In other words, it provides a sufficient input for the forward solver to simulate the data. The forward solver is an operator $\mathbf{g} : \mathfrak{M} \to \mathbb{R}^N$, mapping from the model manifold $\mathfrak{M}$ to the data space. For example, the observed data can be a time series of hydraulic heads at different locations, or a time series of tracer concentrations, as seen in Section 4.3. The model space can describe a field of geological facies in the subsurface (discrete model space) or a field hydraulic properties (continuous model space). The forward operator can be a groundwater flow solver or transport solver. Usually, it solves a set of partial differential equations. The output of the forward solver is deterministic: given the same model, the simulated data are uniquely defined.

The probabilistic solution to the inverse problem is given by:

$$\sigma(\mathbf{m}) = c\rho(\mathbf{m})L(\mathbf{m}; \mathbf{d}^{\text{obs}}), \tag{6.1}$$

with $\sigma(\mathbf{m})$ the posterior probability distribution, c some normalization constant, $\rho(\mathbf{m})$ prior probability distribution, and $L(\mathbf{m}; \mathbf{d}^{\text{obs}})$ the likelihood function. The likelihood function $L(\mathbf{m}; \mathbf{d}^{\text{obs}})$ describes how likely is the model given the observations (it measures the mismatch between the simulated and the observed data) and it depends on the problem at hand (we indicated that the function L uses the observed data with $L(\mathbf{m}; \mathbf{d}^{\text{obs}})$, but we will write $L(\mathbf{m})$ for brevity). The prior probability distribution $\rho(\mathbf{m})$ contains knowledge which is independent of measured data. It is a domain-specific (expert) knowledge about model parameters; for example, constraining models to be generated by a specific geostatistical method. In practice, the normalization constant c does not play a role, as model manifold is approximated using a finite-dimensional space, and solutions can be self-normalized.

The characterization of the posterior distribution $\sigma(\mathbf{m})$ is the goal of the inversion algorithms, and any useful property can be written as a prediction in the following manner:

$$\mu = \int_{\mathfrak{M}} \sigma(\mathbf{m}) f(\mathbf{m}) d\mathbf{m}, \tag{6.2}$$

where μ represents the prediction (expected value) of the quantity of interest which is obtained using function $f(\mathbf{m})$. Typically, Monte Carlo methods aim to sample the posterior, and then use a subset $\mathcal{M} \subset \mathfrak{M}$ to approximate the integral using a sum.

PoPEx + MPS

The Posterior Population Expansion (PoPEx) algorithm (Jäggli et al. 2017; Jäggli et al. 2018) is an adaptive importance sampling technique designed for solving inverse problems in the context of categorical geostatistical fields. In section 4.2, we introduced the stochastic formulation of the inverse problem and the PoPEx approach. Here, we provide a slightly more detailed explanation.

PoPEx is a Monte Carlo sampler aiming to approximate the prediction (6.2) using the self-normalized estimator (5.2) and the weights given by (5.3), see page 89. The main idea of PoPEx is to iteratively expand the set of generated models $\{\mathbf{m}^{(1)}, \mathbf{m}^{(2)}, \ldots, \mathbf{m}^{(k)}\}$ by using information from all previous models, and gradually improving the sampling distribution ϕ. The superscript in brackets indexes the corresponding iteration. If we look more closely at the formula for the weights (5.3), we can notice that they depend on ratio $\rho(\mathbf{m})/\phi(\mathbf{m})$ and on the likelihood $L(\mathbf{m})$. Likelihood is computed by using the forward solver, as it measures mismatch of observed data and the simulated data. The ratio is provided by the PoPEx algorithm.

Each model $\mathbf{m}$ is a (column) vector of d discrete parameters:

$$\mathbf{m} = [m_1, m_2, \ldots, m_d]^\top \in \mathcal{F}^d, \tag{6.3}$$

where we used $\mathcal{F}$ to denote the set of indexes corresponding to different categories (facies): $\mathcal{F} = \{1, 2, \ldots, M\}$, with M the total number of categories. Each of vector element (parameter) m_i is a category (index), taking one discrete value from the set $\mathcal{F}$.

Prior probabilities of parameters are represented by the matrix $\mathbf{Q} \in \mathbb{R}^{d \times M}$ such that $Q_{ij} \in [0, 1]$ and $\sum_{k=1}^{M} Q_{ik} = 1$ for every $i \in \{1, 2, \ldots, d\}$ and $j \in \{1, 2, \ldots, M\}$. Each element Q_{ij} of the matrix describes the probability of the parameter i to be in category j. A possible way to build the $\mathbf{Q}$ matrix at the beginning of PoPEx algorithm, is to run p times the geostatistical simulator and gather the obtained models $\mathbf{m}$ as columns in the matrix $\mathbf{M}^{\text{prior}} \in \mathcal{F}^{d \times p}$. Then for each $i \in \{1, 2, \ldots, d\}$ and $j \in \{1, 2, \ldots, M\}$:

$$Q_{ij} = \frac{1}{p} \sum_{k=1}^{p} \mathbb{1}_j(M_{ik}^{\text{prior}}), \tag{6.4}$$

where $\mathbb{1}_j(x)$ is indicator function defined as: $\mathbb{1}_j(x) = 1$ if $j = x$, otherwise 0.

Similarly to prior probabilities, informed probabilities (closer to posterior but not equivalent to them) are represented by a matrix $\mathbf{P} \in \mathbb{R}^{d \times M}$ such that

$P_{ij} \in [0,1]$ and $\sum_{k=1}^{M} P_{ik} = 1$ for every $i \in \{1,2,\ldots,d\}$ and $j \in \mathcal{F}$. Each element P_{ij} of the matrix describes the probability of the parameter i to be in category j. At the beginning of PoPEx sampling, the matrix $\mathbf{P}$ is not used, and then, it is updated before each new iteration.

The Kullback-Leibler divergence (KLD), also known as relative entropy, has already been introduced (4.1). KLD map will be used to guide the sampling of the new model, and it will be used to impose conditioning data for the geostatistical method. The conditioning data should help to generate more probable models. The KLD map is a vector $\mathbf{h} \in \mathbb{R}^d$ and its elements h_i for each $i \in \{1,2,\ldots,d\}$ are defined as relative entropy from probability distribution $[Q_{i1}, Q_{i2}, \ldots, Q_{iM}]$ to probability distribution $[P_{i1}, P_{i2}, \ldots, P_{iM}]$:

$$h_i = \sum_{j=1}^{M} P_{ij} \log \frac{P_{ij}}{Q_{ij}}. \tag{6.5}$$

The KLD map attributes higher values to locations i, where the most surprising facies come up. The PoPEx idea lies in constraining new models in locations, which correspond to the surprising facies, to sample more efficiently.

PoPEx relies on a conditional sampler, e.g. on a geostatistical method, which is able to produce models from manifold $\mathfrak{M}$, given conditioning data, that is a set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_{n_c}, y_{n_c})\}$ where (x_j, y_j) is a pair of parameter index and facies index for each $j \in \{1,2,\ldots,n_c\}$, where n_c is the number of conditioning points. The conditional sampler, given the conditioning data, $\{(x_1, y_1), (x_2, y_2), \ldots, (x_{n_c}, y_{n_c})\}$ must produce a model $\mathbf{m}$ such that $m_{x_j} = y_j$ for each $j \in \{1,2,\ldots,n_c\}$. Most geostatistical simulators accept conditioning data, often referred to as hard conditioning data, or hard data.

During the first PoPEx iteration, an unconditional model $\mathbf{m}^{(1)}$ is sampled. Now, let us suppose that PoPEx has run for $k \geq 1$ iterations. Before we explain how a new model $\mathbf{m}$ is generated, we need a few definitions. We define the matrix of all generated models:

$$\mathbf{M} = [\mathbf{m}^{(1)}, \mathbf{m}^{(2)}, \ldots, \mathbf{m}^{(k)}]^\top \in \mathcal{F}^{d \times k}. \tag{6.6}$$

The vector of likelihoods contains the corresponding likelihood values:

$$\mathbf{l} = [L(\mathbf{m}^{(1)}), L(\mathbf{m}^{(2)}), \ldots, L(\mathbf{m}^{(k)})]^\top \in \mathbb{R}^k. \tag{6.7}$$

The evaluation of likelihood function requires a forward model run. In AIS scheme, an implementation of weights is needed to adapt the sampling distribution. Different weights are possible, and a most obvious choice is that of normalized likelihoods, therefore we define the vector of weights $\mathbf{s} \in \mathbb{R}^k$ as follows:

$$\mathbf{s} = \frac{\mathbf{l}}{\mathbf{l}^\top \mathbf{1}_k}, \tag{6.8}$$

with $\mathbf{1}_k$ standing for the vectors of ones of length k:

$$\mathbf{1}_k = [1, 1, \ldots, 1]^\top \in \mathbb{R}^k. \tag{6.9}$$

We also need the vector of AIS ratios:

$$\mathbf{r} = \left[\frac{\rho(\mathbf{m}^{(1)})}{\phi_1(\mathbf{m}^{(1)})}, \frac{\rho(\mathbf{m}^{(2)})}{\phi_2(\mathbf{m}^{(2)})}, \ldots, \frac{\rho(\mathbf{m}^{(k)})}{\phi_k(\mathbf{m}^{(k)})}\right]^\top \in \mathbb{R}^k. \tag{6.10}$$

Later, we will explain how the ratios are computed in PoPEx algorithm. We note that $r_1 = 1$, since the first model was unconditional, thus $\phi_1 \equiv \rho$. In practice, $\mathbf{M}, \mathbf{l}, \mathbf{r}$ are expanded during each iteration; only $\mathbf{s}$ needs to be recalculated.

We are ready to explain the generation of a new model. First, the informed probability maps $\mathbf{P}$ are updated:

$$P_{ij} = \sum_{l=1}^{k} \mathbb{1}_j(M_{il}) s_l, \tag{6.11}$$

for $i \in \{1, 2, \ldots, d\}$ and $j \in \{1, 2, \ldots, M\}$. Second, KLD map $\mathbf{h}$ is updated (6.5) and normalized KLD map computed:

$$\tilde{\mathbf{h}} = \frac{\mathbf{h}}{\mathbf{h}^\top \mathbf{1}_d}. \tag{6.12}$$

Third, the number of conditioning points n_c is sampled uniformly from $\{0, 1, \ldots, N_c\}$, where N_c is the maximum number of conditioning points specified by the user. Sometimes, 0 can be drawn, and it implies that non-conditioned model will be sampled. It ensures that innovations can be sampled to avoid bias. Fourth, n_c indexes $(x_1, x_2, \ldots, x_{n_c})$ are drawn (without replacement) from $\{1, 2, ..., d\}$ according to the distribution $\tilde{\mathbf{h}}$. Now, n_c model indexes $(t_1, t_2, \ldots, t_{n_c})$ are sampled (with replacement) from $\{1, 2, \ldots, k\}$ according to the distribution s. Fifth, we set $y_i = m_{x_i}^{t_i}$ for $i \in \{1, 2, \ldots, n_c\}$. Finally, a new model $\mathbf{m}^{(k+1)}$ is generated by the conditioning sampler with conditioning data $\{(x_1, y_1), (x_2, y_2) \ldots, (x_{n_c}, y_{n_c})\}$, and the forward solver is called $\mathbf{g}(\mathbf{m})$ to obtain the likelihood l_{k+1}. The ratio r_{k+1} is gien by the following formula (Jäggli et al. 2018):

$$r_{k+1} = \prod_{i=1}^{n_c} \frac{Q_{x_j y_j}}{P_{x_j y_j}}. \tag{6.13}$$

If no conditioning data was used ($n_c = 0$), then $r_{k+1} = 1$.

Once the models are generated (PoPEx stops after a predefined number of steps by the user), PoPEx can use weights for generating predictions. The algorithm for computing weights was described in the previous chapter. The parallelized version is based on asynchronuous worker processes and is explained in Jäggli et al. (2018).

In this study and in previous ones (Dagasan et al. 2020; Jäggli et al. 2017; Jäggli et al. 2018), PoPEx was coupled with the Direct Sampling (DS) MPS algorithm to generate the categorical fields. More precisely, we use the DeeSse implementation with multi-resolution features (Straubhaar et al. 2020). The multi-resolution capability (Gaussian pyramids) is a technique allowing to improve the reproduction of patterns at different scales.

ESMDA + DS pyramid

The second inversion method that we will compare is the one proposed by Lam et al. (2020). It is based on the ensemble smoother with multiple data assimilation (ESMDA) coupled with DS with Gaussian pyramids. ESMDA (Emerick and Reynolds 2013) runs for a predefined number of steps N_a (parameter given by the user, also known as number of data assimilations), and at each iteration $k \in \{1, 2, \dots, N_a\}$ the ensemble N_e of models $\{\mathbf{m}_1^k, \mathbf{m}_2^k, \dots \mathbf{m}_{N_e}^k\}$ is updated to $\{\mathbf{m}_1^{k+1}, \mathbf{m}_2^{k+1}, \dots \mathbf{m}_{N_e}^{k+1}\}$. We use subscript here for the model index and superscript for the iteration index. The index 1 corresponds to the initial (prior) ensemble, and the ensemble after N_a data assimilations will have index $N_a + 1$. We based the algorithmic implementation of the method on the paper by Emerick (2016).

Contrary to PoPEx, in the ESMDA framework, a model is a vector of real values (not discrete): $\mathbf{m}_i^k \in \mathbb{R}^{N_m}$, for all $i \in \{1, 2, \dots, N_e\}$ and $k \in \{1, 2, \dots, N_a + 1\}$, it contains N_m parameters which will be assimilated. The observation vector $\mathbf{d}_{\text{obs}} \in \mathbb{R}^{N_d}$ contains N_d observation points. These vectors are considered column vectors. To obtain vector elements, we will use bracket notation. For example, to explicit model vector, we have: $\mathbf{m}_i^k = \left[\mathbf{m}_i^k[1], \mathbf{m}_i^k[2], \dots, \mathbf{m}_i^k[N_m]\right]^\top$.

The equation for model update in ESDMA is:

$$\mathbf{m}_i^{k+1} = \mathbf{m}_i^k + \mathbf{C}_{\text{MD}}^k \left(\mathbf{C}_{\text{DD}}^k + \alpha_k \mathbf{C}_{\text{err}}\right)^{-1} \left(\mathbf{d}_{\text{obs},i}^k - \mathbf{g}(\mathbf{m}_i^k)\right) \tag{6.14}$$

with $\mathbf{g}$ the forward operator, $\mathbf{C}_{\text{MD}}^k \in \mathbb{R}^{N_e \times N_d}$ cross-covariance matrix between model $\mathbf{m}_i^k \in \mathbb{R}^{N_m}$ and simulated data $\mathbf{g}(\mathbf{m}_i^k) \in \mathbb{R}^{N_d}$; $\mathbf{C}_{\text{DD}}^k \in \mathbb{R}^{N_d \times N_d}$ autocovariance matrix of simulated data $\mathbf{g}(\mathbf{m}_i^k)$; $\mathbf{C}_{\text{err}} \in \mathbb{R}^{N_d \times N_d}$ covariance of observed data, α_k inflating factor, and $\mathbf{d}_{\text{obs},i}^k$ perturbed observation data. The inflating factors should satisfy the condition $\sum_{k=1}^{N_a} 1/\alpha_k = 1$, and $\alpha_k = N_a$ for all $k \in \{1, 2, \dots, N_a\}$ is a valid choice, and it will be used in our study. At each iteration, a new set of perturbed observation data is obtained in the following way: $\mathbf{d}_{\text{obs},i}^k = \mathbf{d}_{\text{obs}} + \sqrt{\alpha_k}\mathbf{C}_{\text{err}}^{1/2}\mathbf{z}_i^k$, where $\mathbf{z}_i^k \sim \mathcal{N}(0, \mathbf{I}_{N_d})$

According to Emerick (2016), we can rewrite equation (6.14) in matrix form. The matrices are assembled from column vectors as follows:

$$\mathbf{M}^k = \left[\mathbf{m}_1^k, \mathbf{m}_2^k, \dots, \mathbf{m}_{N_e}^k\right] \in \mathbb{R}^{N_m \times N_e}, \tag{6.15}$$

and more explicitly:

$$\mathbf{M}^k = \begin{bmatrix} \mathbf{m}_1^k[1] & \mathbf{m}_2^k[1] & \dots & \mathbf{m}_{N_e}^k[1] \\ \mathbf{m}_1^k[2] & \mathbf{m}_2^k[2] & \dots & \mathbf{m}_{N_e}^k[2] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{m}_1^k[N_m] & \mathbf{m}_2^k[N_m] & \dots & \mathbf{m}_{N_e}^k[N_m] \end{bmatrix}. \tag{6.16}$$

This matrix assembles all models from iteration k and puts them as columns. Rows correspond to the ensembles of specific parameters. Let us now define more matrices:

$$\mathbf{\Delta M}^k = \left[\mathbf{m}_1^k - \widehat{\mathbf{m}}^k, \mathbf{m}_2^k - \widehat{\mathbf{m}}^k, \dots, \mathbf{m}_{N_e}^k - \widehat{\mathbf{m}}^k\right] \in \mathbb{R}^{N_m \times N_e}, \tag{6.17}$$

where $\widehat{\mathbf{m}}^k = 1/N_e \sum_{i=1}^{N_e} \mathbf{m}_i^k$. Now the matrix of simulated data:

$$\mathbf{D}^k = \left[\mathbf{d}_1^k, \mathbf{d}_2^k, \dots, \mathbf{d}_{N_e}^k\right] \in \mathbb{R}^{N_d \times N_e}, \tag{6.18}$$

where $\mathbf{d}_i^k = \mathbf{g}(\mathbf{m}_i^k)$ for $i \in \{1, 2, \dots, N_e\}$. Its delta matrix is given by:

$$\mathbf{\Delta D}^k = \left[\mathbf{d}_1^k - \widehat{\mathbf{d}}^k, \mathbf{d}_2^k - \widehat{\mathbf{d}}^k, \dots, \mathbf{d}_{N_e}^k - \widehat{\mathbf{d}}^k\right], \tag{6.19}$$

where $\widehat{\mathbf{d}}^k = 1/N_e \sum_{i=1}^{N_e} \mathbf{d}_i^k$. Finally, the observed perturbation matrix:

$$\mathbf{D}_{\text{obs}}^k = \left[\mathbf{d}_{\text{obs},1}^k, \mathbf{d}_{\text{obs},2}^k, \dots, \mathbf{d}_{\text{obs},N_e}^k\right]. \tag{6.20}$$

The matrix equation for the data assimilation is as follows:

$$\mathbf{M}^{k+1} = \mathbf{M}^k + \mathbf{\Delta M}^k (\mathbf{\Delta D}^k)^\top \left(\mathbf{\Delta D}^k (\mathbf{\Delta D}^k)^\top + \alpha_k \mathbf{C}_{err}\right)^{-1} \left(\mathbf{D}_{\text{obs}}^k - \mathbf{D}^k\right). \tag{6.21}$$

We note here that the equation involves the inverse of the matrix

$$\mathbf{C} = \left(\mathbf{\Delta D}^k (\mathbf{\Delta D}^k)^\top + \alpha_k \mathbf{C}_{err}\right). \tag{6.22}$$

The matrix $\mathbf{C}$ is of size $N_d \times N_d$, and in the case $N_d \gg N_e$, Emerick (2016) proposes to use the subspace inversion procedure. In our case, it is possible to use directly pseudo-inverse of $\mathbf{C}$, which allows avoiding numerical problems if the matrix is poorly conditioned but is still computationally efficient.

Ensemble Kalman filters and ensemble smoothers are affected by spurious correlations, and we will use tapering technique ("covariance localization") as described in the article of Lam et al. (2020). We will therefore use the tapering matrix $\mathbf{T} \in \mathbb{R}^{N_m \times N_d}$ in equation (6.21). The tapering matrix is multiplied element-wise (Hadamard product, denoted with $\circ$)with the Kalman gain term.

The final update equation will look like this:

$$\mathbf{M}^{k+1} = \mathbf{M}^k + \left(\mathbf{T} \circ \left(\mathbf{\Delta M}^k (\mathbf{\Delta D}^k)^\top \mathbf{C}^+\right)\right) \left(\mathbf{D}_{\text{obs}}^k - \mathbf{D}^k\right), \tag{6.23}$$

where $\mathbf{C}^+$ is (Moore-Penrose) pseudo-inverse of matrix $\mathbf{C}$ computed by means of picking only significant values in SVD, as implemented in scipy package by Virtanen et al. (2020).

The tapering approach is based on the fifth-order correlation function with compact support (Gaspari and Cohn 1999). Before we define the tapering matrix, let us first define m_{ij} for $i \in \{1, 2, \ldots, N_m\}$ and $j \in \{1, 2, \ldots, N_d\}$:

$$m_{ij} = \frac{d(\mathbf{m}[i], \mathbf{d}^{\text{obs}}[j])}{r}, \tag{6.24}$$

where d is a distance function, which measures the distance in space between model parameter i and observation j; r the is range, a distance considered that observation has a significant influence on parameter. If the range is small, it means that observations spatially far from parameters should not influence them much.

Now the tapering matrix:

$$T_{ij} = \begin{cases} -\frac{1}{4}m_{ij}^5 + \frac{1}{2}m_{ij}^4 + \frac{5}{8}m_{ij}^3 - \frac{5}{3}m_{ij}^2 + 1 & \text{if } m_{ij} < 1 \\ \frac{1}{12}m_{ij}^5 - \frac{1}{2}m_{ij}^4 + \frac{5}{8}m_{ij}^3 + \frac{5}{3}m_{ij}^2 - 5m_{ij} + 4 - \frac{2}{3}\frac{1}{m_{ij}} & \text{if } 1 < m_{ij} < 2 \\ 0 & \text{otherwise,} \end{cases} \tag{6.25}$$

for all $i \in \{1, 2, \ldots, N_m\}$ and $j \in \{1, 2, \ldots, N_d\}$.

So far, the described method concerns matching data represented by continuous values. The main novelty of the approach proposed by Lam et al. (2020) is the way of conditioning categorical simulations with continuous variables. Therefore, the ESMDA procedure is a standard one, but the data that is assimilated is used to condition categorical simulations. In this subsection, we will review briefly how it is done. We need two ingredients: a procedure of generating an initial ensemble of models, which is a vector of continuous parameters, and a procedure to generate a categorical models based on such a vector.

Coupling DS and ESMDA

The ensemble of models $\mathbf{M}^1 = [\mathbf{m}_1^1, \mathbf{m}_2^1, \ldots, \mathbf{m}_{N_e}^1]$ is generated using the following steps.

We use the multi-resolution option of the DeeSse code (Straubhaar et al. 2020) to generate unconditional realizations. The realizations are categorical (of two categories) but they also include low resolution continuous images over the same grid. These continuous values are simulated jointly with the categorical values. The link between the continuous and categorical variables is established on the training image using Gaussian kernels to blur and represent the field at a lower resolution (in practice, multiple pyramid levels are used). At the coarse resolution, a fraction f of the total number of cells is sampled to obtain an ensemble

of pyramid values (now continuous) at fixed locations $\{\mathbf{p}_1^1, \mathbf{p}_2^1, \ldots, \mathbf{p}_{N_e}^1\}$, with $\mathbf{p}_i^k \in \mathbb{R}^{N_p}$, where k is iteration index, and i is ensemble member index. $\mathbf{p}_i^k[j]$ represents a Gaussian pyramid value at a location with index j. While it would be possible to use directly $\mathbf{p}_i^k$ vectors in the ESMDA procedure, it is not a good idea, because these parameter distributions are not necessarily Gaussian and ESMDA performance will be hindered. Therefore, Lam et al. (2020) suggests to use normal score transform, as proposed in the study of Zhou et al. (2011).

The normal score transfer function is constructed for each parameter in the vector $\mathbf{p}_i^1[j]$ and is kept fixed for the entire data assimilation process. Let F_j for all $j \in \{1, 2, \ldots, N_m\}$ be the cumulative distribution function (CDF) deduced from the ensemble $\{\mathbf{p}_1^1[j], \mathbf{p}_2^1[j], \ldots, \mathbf{p}_{N_e}^1[j]\}$. For each pyramid location j corresponding F_j is computed with its inverse F_j^{-1} and they are stored. Now the direct normal score transform is defined:

$$\Phi_i^{\text{direct}}(x) = G^{-1}(F_j(x)), \tag{6.26}$$

where G^{-1} is the inverse of normal CDF. The normal score back transform is given by:

$$\Phi_i^{\text{back}}(x) = F_j^{-1}(G(x)), \tag{6.27}$$

where F_j^{-1} is inverse of the pyramid CDF, and G stands for normal CDF.

Finally, each parameter vector $\mathbf{m}_i^1 = \left[\mathbf{m}_i^1[1], \mathbf{m_i}^1[2], \ldots, \mathbf{m}_i^1[N_e]\right]^\top$ initial ensemble can be obtained:

$$\mathbf{m}_i^1[j] = \Phi_j^{\text{direct}}(\mathbf{p}_i^1[j]) \tag{6.28}$$

for all $j \in \{1, 2, \ldots, N_m\}$ and $i \in \{1, 2, \ldots, N_e\}$. To transfer the parameter vector $\mathbf{m}_i^k$ into pyramid conditioning data we employ the back transform:

$$\mathbf{p}_i^k[j] = \Phi_j^{\text{back}}(\mathbf{m}_i^k[j]), \tag{6.29}$$

for $j \in \{1, 2, \ldots, N_m\}$. The set of pyramids can now be used in the DeeSse implementation to obtain the realizations by conditioning the simulations at the coarse resolution. It is important to note that the same simulation seed must be used at each iteration for the given realization index i. It ensures that the realizations are gradually improved by the inversion. Otherwise, the stochastic nature of the MPS simulation would make it impossible. Before every forward model call, the model parameters are converted to pyramid values and the subsequent realization generated. The details of the multi-resolution Direct Sampling implementation are given in Straubhaar et al. (2020), and the details about the conditioning of Gaussian pyramids are in Lam et al. (2020).

DREAM-ZS + WGAN

The third inversion method is DREAM-ZS used with a Wasserstein Generative Adversarial Net (WGAN). It was proposed by Laloy et al. (2018), and we use it only with a slight modification: with Wasserstein GAN, instead with Spatial GAN.

DREAM-ZS

DREAM-ZS is a modified Metropolis sampler, sampling multiple chains which exchange information using an archive $\mathcal{Z}$ of past models (Laloy et al. 2018, 2017). Metropolis samplers generate proposals and accept them if their likelihood is higher, or with a probability if it is lower. In a standard Metropolis sampler, chains would not communicate with each other, which makes it easily parallelizable, but requires removing outlier trajectories; this results in a slower convergence. DREAM-ZS provides a way to allow efficient parallelization and communication within the chains; hence avoids necessity to remove outlier chains. Usually, the sampler is run unless a convergence criterion is satisfied (for example (Gelman and Rubin 1992)) but in practical cases with large number of data, the convergence might not be achieved in a reasonable number of iterations (Laloy et al. 2018). Therefore, we simply run here N_c chains during a predefined number of iterations T and use the two last recorded samples from each chain to form the posterior. Samples are recorded every K iterations. Our implementation uses a Wasserstein GAN (WGAN), instead of a spatial GAN (SGAN) as it was suggested by Laloy et al. (2018), because WGAN are supposed to be more stable, and easier to train for different training data sets. More details on our WGAN setup are given in the next subsection. The details of our implementation are based on the MT-DREAM-ZS paper Laloy and Vrugt (2012) but we do not use multiple try (MT) technique. Our implementation essentially corresponds to MT-DREAM-ZS with one trial. The algorithm for computation of crossover values is based on the work of Vrugt et al. (2009).

The parameter space is the latent space of the GAN, $\mathbf{x}$ is the parameter vector of length d: $\mathbf{x} \in \mathbb{R}^d$. We will use $\mathcal{Z}$ to denote the archive, which is the collection of past models used to create new proposals in the Markov chain; the archive is updated every K iteration. The posterior should be formed by taking samples from $\mathcal{Z}$ and ignoring initial and burn-in samples. In our setting, we propose to take last $2N_c$ samples from $\mathcal{Z}$, where N_c is the number of chains. It means that two last archived samples per each chain are conserved for posterior.

The initial archive $\mathcal{Z}$ is composed of N_p (number of prior samples in the archive) random normal vectors:

$$\mathcal{Z} = \left\{ \mathbf{x}_l : \mathbf{x}_l \sim \mathcal{N}(0, \mathbf{I}^d), l \in \{1, 2, \ldots, N_p\} \right\}, \tag{6.30}$$

with $\mathbf{I}_d$ identity matrix of size $d \times d$, and $\mathcal{N}$ normal multivariate distribution. We used subscript to index different chains, and not vector elements. Similarly, the initial vector in each chain is sampled from $\mathcal{N}(0, \mathbf{I}_d)$.

In each chain $i \in \{1, 2, \ldots, N_c\}$, for all $t < T$, (t is the iteration index) a transition from the current point $\mathbf{x}_i$ to a new point $\mathbf{x}'_i$ is proposed. There are two ways to generate a proposal point in DREAM-ZS: either by the parallel update, or by the snooker update. The snooker update is applied with certain frequency (f_s), otherwise parallel update is applied. The proposed point is

always accepted if its likelihood $L(\mathbf{x}_i')$ is higher than $L(\mathbf{x}_i)$, otherwise it is accepted with probability $L(\mathbf{x}_i')/L(\mathbf{x}_i)$. If the point is accepted, we set $\mathbf{x}_i = \mathbf{x}_i'$, otherwise the state of the chain remains unchanged. Every K iterations the archive $\mathcal{Z}$ is updated:

$$\mathcal{Z} = \mathcal{Z} \cup \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_c}\}. \tag{6.31}$$

The snooker update was described by ter Braak and Vrugt (2008). It consists in drawing 3 vectors $\mathbf{z}$, $\mathbf{z}_1$, $\mathbf{z}_2$ from the archive $\mathcal{Z}$ without replacement and projecting the difference $\mathbf{z}_1 - \mathbf{z}_2$ on the line $\mathbf{x} - \mathbf{z}$ (where $\mathbf{x}$ is the current state of the chain), to get the projection $\mathbf{w}$. The projection is then used to generate a proposal:

$$\mathbf{x}' = \mathbf{x} + \gamma_s \mathbf{w}, \tag{6.32}$$

where γ_s is drawn from $\mathscr{U}(1.2, 2.2)$, being the uniform distribution in the interval $[1.2, 2.2]$. The snooker update can be summarized using an algorithm:

SNOOKER-UPDATE$(\mathbf{x}, \mathcal{Z})$
1 Sample $\mathbf{z}$, $\mathbf{z}_1$, $\mathbf{z}_2$ from $\mathcal{Z}$ without replacement
2 $\mathbf{w} = \frac{(\mathbf{x}-\mathbf{z})(\mathbf{x}-\mathbf{z})^\top(\mathbf{z}_1-\mathbf{z}_2)}{(\mathbf{x}-\mathbf{z})^\top(\mathbf{x}-\mathbf{z})}$
3 Draw $\gamma_s \sim \mathscr{U}(1.2, 2.2)$
4 $\mathbf{x}' = \mathbf{x} + \gamma_s \mathbf{w}$
5 **return** $\mathbf{x}'$

The parallel update is given by the following formula:

$$\mathbf{x}' = \mathbf{x} + \mathbf{c} \odot (\gamma(\delta, d') [(\mathbf{1}_d + \mathbf{e}) \odot (\mathbf{z}_1 + \mathbf{z}_2)] + \boldsymbol{\epsilon}), \tag{6.33}$$

where we used several new symbols, which we will now explain. $\mathbf{c} \in \mathbb{R}^d$ is a vector composed of 0 and 1, indicating (with 1), which dimensions of $\mathbf{x}$ will be updated or not (with 0). The procedure for calculating the vector $\mathbf{c}$ is given in the next paragraph. The Hadamard product (piecewise product) is denoted with $\odot$. d' is the number of updated dimensions: $d' = \mathbf{c}^\top \mathbf{c}$. The jump size γ is given by (Laloy and Vrugt 2012):

$$\gamma(\delta, d') = \begin{cases} 1 & \text{if } U < 0.2, \, U \text{ drawn from } \mathscr{U}(0,1), \\ 2.38/\sqrt{2\delta d'} & \text{otherwise.} \end{cases} \tag{6.34}$$

Originally, it was suggested that every fifth iteration the jump size is set to 1, in our implementation it is set every fifth iteration on average. $\mathbf{1}_d$ stands for a vector of all ones of length d: $\mathbf{1}_d = [1, 1, \ldots, 1]^\top \in \mathbb{R}^d$. The vector $\mathbf{e}$ is sampled from the uniform distribution: $\mathbf{e} \sim \mathscr{U}^d(-b, b)$. The vector $\boldsymbol{\epsilon}$ is sampled from the normal distribution: $\mathscr{N}(0, b^\star \mathbf{I}_d)$, with $b^\star$ controlling the variance. Finally, δ, $\mathbf{z}_1$ and $\mathbf{z}_2$ are obtained in the following manner: first, δ is uniformly drawn

from $\{1, 2, \ldots, \delta_{\max}\}$, where $\delta_{\max}$ controls the maximum number of sampled pairs. Second, 2δ vectors are drawn without replacement from $\mathcal{Z}$, let us call them: $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_\delta, \mathbf{s}_{\delta+1}, \ldots, \mathbf{s}_{2\delta}$. The sampled vectors are averaged:

$$\mathbf{z}_1 = \frac{1}{\delta} \sum_{i=1}^{\delta} \mathbf{s}_i, \qquad \mathbf{z}_2 = \frac{1}{\delta} \sum_{i=\delta+1}^{2\delta} \mathbf{s}_i. \tag{6.35}$$

The determination of updated dimensions is performed for each dimension, therefore for $i \in \{1, 2, \ldots, d\}$ a value is $U \sim \mathscr{U}(0, 1)$ is drawn and elements of vector $\mathbf{c}$ are set using the rule:

$$c_i = \begin{cases} 0 & \text{if } U \leq 1 - CR \\ 1 & \text{otherwise} \end{cases} \tag{6.36}$$

we used CR for cross-over probability value. For the moment, let us suppose that CR is given. We will detail the implementation of CR calculation after stating the UPDATED-DIMENSIONS and PARALLEL-UPDATE algorithms. The procedure for determining the vector in the form of an algorithm:

UPDATED-DIMENSIONS(CR))
1 $\mathbf{c} = \mathbf{1}^d$
2 **for** $i = 1$ **to** d
3 draw $U \sim \mathscr{U}(0, 1)$
4 **if** $U \leq 1 - CR$
5 $c_i = 0$
6 **return** $\mathbf{c}$

Now we are ready to write the parallel update algorithm:

PARALLEL-UPDATE($\mathbf{x}, \mathcal{Z}, CR$)
1 Draw δ uniformly from $\{1, 2, \ldots, \delta_{\max}\}$
2 Draw 2δ samples from $\mathcal{Z}$ without replacement: $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_\delta, \mathbf{s}_{\delta+1}, \ldots, \mathbf{s}_{2\delta}$
3 $\mathbf{z}_1 = \frac{1}{\delta} \sum_{i=1}^{\delta} \mathbf{s}_i$
4 $\mathbf{z}_2 = \frac{1}{\delta} \sum_{i=\delta+1}^{2\delta} \mathbf{s}_i$
5 draw $\boldsymbol{\epsilon} \sim \mathscr{N}(0, b^\star \mathbf{I}_d)$
6 draw $\mathbf{e} \sim \mathscr{U}^d(-b, b)$
7 $\mathbf{c}$ = UPDATED-DIMENSIONS(CR)
8 $d' = \mathbf{c}^\top \mathbf{c}$
9 $\mathbf{x}' = \mathbf{x} + \mathbf{c} \odot (\gamma(\delta, d') [(\mathbf{1}_d + \mathbf{e}) \odot (\mathbf{z}_1 + \mathbf{z}_2)] + \boldsymbol{\epsilon})$
10 **return** $\mathbf{x}'$

The final piece of the DREAM-ZS algorithm is the implementation of crossover values, which has two ingredients: determination of CR value for each chain,

and the *CR* distribution improvement. Unlike Vrugt et al. (2009), we improve the *CR* distribution at every iteration until the end of DREAM-ZS algorithm.

The determination of *CR* value for the chain i proceeds as follows. Suppose that we have a probability vector $p \in \mathbb{R}^{n_{CR}}$ such that $p_m \in [0,1]$ and $\sum_{m=1}^{n_{CR}} p_m = 1$. The value v_i is drawn from a categorical distribution with possible values $\{1, 2, \ldots, n_{CR}\}$ and corresponding probabilities $\mathbf{p}$. Let us use $\mathscr{M}(\{1, 2, .., n_{CR}\}, \mathbf{p})$ to denote such a categorical distribution. The corresponding *CR* value is set as: $CR = 1/v_i$.

Now, we need a way to update the $\mathbf{p}$ vector. The initial values of elements for the vector are $p_m = 1/n_{CR}$ for $m \in \{1, 2, \ldots, n_{CR}\}$ and these values are recalculated after each DREAM-ZS iteration. Let $\mathbf{v} \in \{1, 2, \ldots, n_{CR}\}^{N_c}$ be the vector with the sampled *CR* values for each chain. Let us define vectors $\boldsymbol{\Delta}, \mathbf{L} \in \mathbb{R}^{n_{CR}}$, initialized with zero vectors, whose elements are updated as follows:

$$\Delta_m = \Delta_m + \sum_{i=1}^{Nc} \mathbb{1}_m(v_i) \sum_{j=1}^{d} \left((\mathbf{x}'_i[j] - \mathbf{x}_i[j])^2 / r_j^2 \right), \quad m \in \{1, 2, \ldots n_{CR}\},$$

with $r_j^2 = \mathrm{Var}(\{\mathbf{x}_1^t[j], \mathbf{x}_2^t[j], \ldots, \mathbf{x}_{Nc}^t[j]\})$ the variance of the parameter j of the state vector among all chains. The square brackets serve to obtain elements of a vector $\mathbf{x}$. The vector $\mathbf{L}$ counts how many times each *CR* value was drawn:

$$L_m = L_m + \sum_{i=1}^{Nc} \mathbb{1}_m(v_i), \quad m \in \{1, 2, \ldots n_{CR}\}. \tag{6.37}$$

Finally, we can state the update of the vector of probabilities:

$$p_m = \frac{\Delta_m / L_m}{\sum_{j=1}^{n_{CR}} (\Delta_j / L_j)}, \quad m \in \{1, 2, \ldots, n_{CR}\}. \tag{6.38}$$

Given all the elements, we are ready to summarize our DREAM-ZS implementation, where GENERATE is the function generating a model from the latent vector (for example, GAN implementation):

DREAM-ZS(T, N_c, K, n_{CR})

Generate prior $\mathcal{Z}$
parallel for $i = 1$ **to** N_c
    Sample initial vector $\mathbf{x}_i \sim \mathcal{N}(0, \mathbf{I}^d)$
    $\mathbf{m}_i$ = GENERATE($\mathbf{x}_i$)
    $\mathbf{p} = \mathbf{1}_{n_{CR}}/n_{CR}$
    $\Delta_m = 0$, $L_m = 0$ for $m = 1$ **to** n_{CR}
for $t = 1$ **to** T
    // decide the kind of update for all chains
    Draw $U_1 \sim \mathscr{U}[0, 1]$
    if $U_1 < f_s$ and $t \neq 1$
        UPDATE = SNOOKER-UPDATE
        $\mathbf{v} = n_{CR}\mathbf{1}_{N_c}$
    else
        UPDATE = PARALLEL-UPDATE
        sample $\mathbf{v} \sim \mathscr{M}^{N_c}(\{1, 2, .., n_{CR}\}, \mathbf{p})$
    parallel for $i = 1$ **to** N_c
        $CR = 1/v_i$
        $\mathbf{x}'_i$ = UPDATE($\mathbf{x}_i, \mathcal{Z}, CR$)
        $\mathbf{m}'$ = GENERATE($\mathbf{x}'_i$)
        Draw $U_2 \sim \mathscr{U}[0, 1]$
        if $U_2 < L(\mathbf{m}'_i)/L(\mathbf{m}_i)$
            $\mathbf{x}_i = \mathbf{x}'_i$
            $\mathbf{m}_i = \mathbf{m}'_i$
    for $m = 1$ **to** n_{CR}
        $\Delta_m = \Delta_m + \sum_{i=1}^{N_c} \mathbb{1}_m(v_i) \sum_{j=1}^{d} \left((\mathbf{x}_i[j] - \mathbf{x}_i^{t-1}[j])^2/r_j^2\right)$
        $L_m = L_m + \sum_{i=1}^{N_c} \mathbb{1}_m(v_i)$
        $\mathbf{p}[m] = \frac{\Delta_m/L_m}{\sum_{j=1}^{n_{CR}}(\Delta_j/L_j)}$
    if $t \bmod K = 0$
        $\mathcal{Z} = \mathcal{Z} \cup \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_c}\}$

WGAN

Generative adversarial neural networks (GAN) can learn a complex mapping between a latent space and the space of two-dimensional images (Goodfellow et al. 2016). In this work, we decided to use the Wasserstein GAN (Arjovsky et al. 2017) with gradient penalty term, which is claimed to be robust for changing architecture of the net (Gulrajani et al. 2017). GANs are composed of two neural networks: a critic (discriminator) and a generator. The generator maps the latent space vectors to the image space. The critic is given the output from the generator or real images (the images from the training set). The goal of the generator is to fool the critic, so that it cannot distinguish the generated images from the images of the training set. Typically, for an epoch (GAN training iteration), critic is optimized several times after a single generator training.

Table 6.1: Layers of the convolutional neural network used for generator

layer type	kernel	stride	padding	output shape
Input				50
2D transp. conv.	4×4	1×1	0×0	
batch norm, ReLU				$1024 \times 4 \times 4$
2D transp. conv.	4×4	2×2	1×1	
batch norm, ReLU				$512 \times 8 \times 8$
2D transp. conv.	4×4	2×2	1×1	
batch norm, ReLU				$256 \times 16 \times 16$
2D transp. conv.	4×4	2×2	1×1	
batch norm, ReLU				$128 \times 32 \times 32$
2D transp. conv.	4×4	2×2	1×1	
batch norm, ReLU				$64 \times 64 \times 64$
2D transp. conv.	4×4	1×1	0×0	$1 \times 128 \times 128$
Tanh				$1 \times 128 \times 128$

In our case, the latent space has d dimensions and the images represent the geology. While the generated images (GAN output) have values between $[-1, 1]$, they can be converted to binary images by applying a threshold (0). However, the threshold is not applied when evaluating the likelihood of the model. Instead, the physical parameters are linearly transformed from the pixel values, with -1 and 1 corresponding to the exact values according to facies. The training set contains all the possible extractions from the TI of the size 128×128. The training batch size is 64, the learning rate 1×10^{-4}, the ADAM optimizer was used with beta parameters: 0.5 and 0.999. There are 5 critic iterations per generator iteration, and the lambda term for the gradient penalty was set to 10. The architectures of the generator and the critic are shown in Tab. 6.1, Tab. 6.2, respectively.

6.3 Test case

The inverse problem

We consider a pumping test in a confined aquifer of thickness 10 m. At the beginning of the test, the hydraulic heads are uniform and constant at 0 m. Water is pumped with a constant discharge rate of $0.08\,\mathrm{m}^3/\mathrm{s}$ during 2 h. The hydraulic heads are recorded in the pumping well and nine piezometers in the vicinity of the pumping well (Fig. 6.1 and Tab. 6.3). The hydraulic heads are recorded every 100 s, so that 72 measurements are available at each of the 10 locations, which makes up for a total of $N = 720$ measurement points (Fig. 6.2).

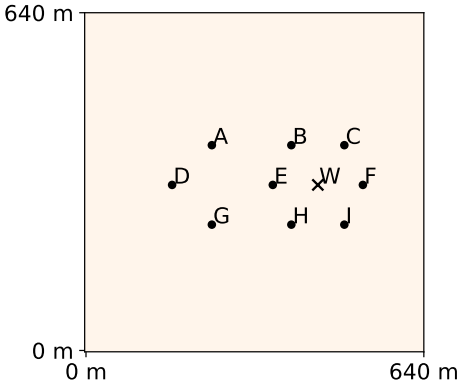

Figure 6.1: Position of the pumping well (W) and nine piezometers (A-I) in the pumping test.

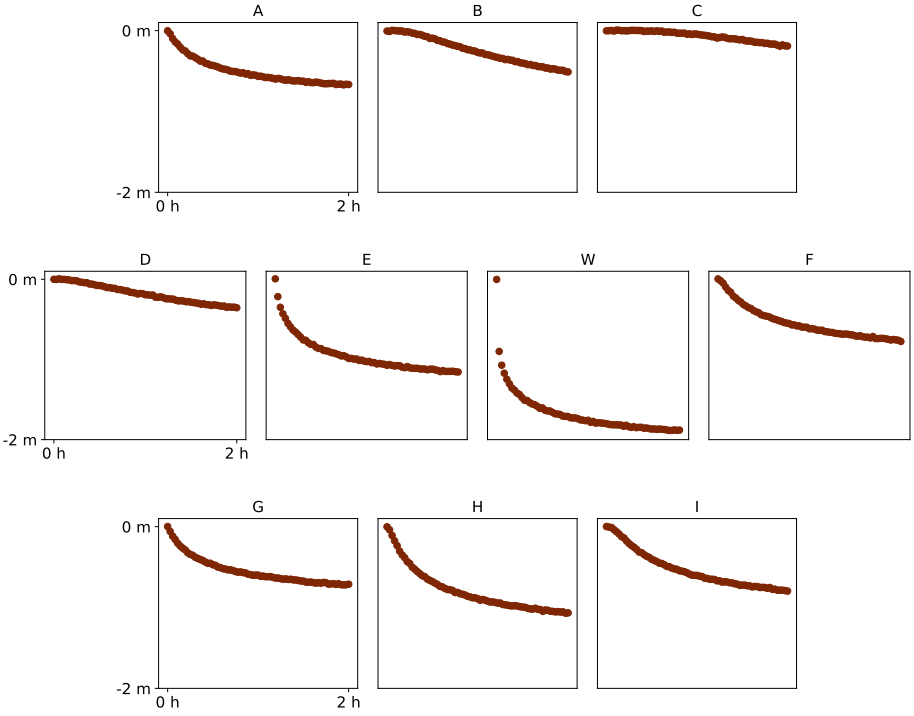

Figure 6.2: Time series of hydraulic heads recorded at nine piezometers (A-I) and the pumping well (W).

Table 6.2: Layers of the convolutional neural network used for critic

layer type	kernel	stride	padding	output shape
Input				$1 \times 128 \times 128$
2D conv.	4×4	2×2	1	
instance norm, leakyReLU				$64 \times 64 \times 64$
2D conv.	4×4	2×2	1×1	
instance norm, leakyReLU				$128 \times 32 \times 32$
2D conv.	4×4	2×2	1×1	
instance norm, leakyReLU				$256 \times 16 \times 16$
2D conv.	4×4	2×2	1×1	
instance norm, leakyReLU				$512 \times 8 \times 8$
2D conv.	4×4	2×2	1×1	
instance norm, leakyReLU				$1024 \times 4 \times 4$
2D conv.	4×4	2×2	1×1	1

x (m)	y (m)	column index	row index	label
242.5	392.5	48	78	A
392.5	392.5	78	78	B
492.5	392.5	98	78	C
167.5	317.5	33	63	D
357.5	317.5	71	63	E
442.5	317.5	88	63	W
527.5	317.5	105	63	F
242.5	242.5	48	48	G
392.5	242.5	78	48	H
492.5	242.5	98	48	I

Table 6.3: Positions of piezometers (x, y coordinates), corresponding columns and row indexes, and labels.

The reference set-up

The data presented in the previous section were obtained from a synthetic set-up. We will refer to it as the reference. It is not the solution of the inverse problem framed in a probabilistic manner. It is rather a model which has a very high likelihood, given the data. The domain has an extension of 640 m by 640 m. The petrophysical parameters are modeled using a categorical 2D field with two geological facies: a permeable (channels or ellipsoidal deposits, labeled with 1) and a less permeable matrix (labeled with 0). The area is discretized using a regular grid with cells of size 5 m by 5 m, thus the grid contains 128 by 128 cells. The two geological facies have constant hydrogeological parameters (Tab. 6.4). The boundary conditions are constant head values at all edges,

	less permeable (0)	more permeable (1)
hydraulic conductivity (m/s)	1×10^{-4}	1×10^{-2}
specific storage (m^{-1})	5×10^{-4}	5×10^{-5}
porosity	0.4	0.3

Table 6.4: Hydrogeological parameters of different geological facies considered in the study.

equal to 0 m, and at the beginning of the pumping test, hydraulic charge equals to 0 m everywhere in the domain. The reference field was created using the DeeSse software, which is an implementation of Direct Sampling algorithm with pyramids (Straubhaar et al. 2020). An extended image of a channelized aquifer was used as the training image (Zahner et al. 2016). We chose to run the DeeSse in the DSBC mode (e.g. using Direct Sampling Best Candidate, see Chapter 3). It boils down to choosing a threshold of 0 in the standard Direct Sampling, or very small close to 0 if the software does not allow for non-positive input. The maximal scan fraction was set to 0.01 and the number of neighboring nodes to 40. Two pyramid levels were used, with the reduction by 2 in each direction at every level. The groundwater flow was simulated using the flopy python package (Bakker et al. 2016), which is a wrapper for the Modflow software (Hughes et al. 2017). To emulate the measurement error, the obtained values of hydraulic heads were corrupted with a gaussian noise with mean 0 m and standard deviation 0.005 m. These data will be used as input for the different inversion procedures.

To evaluate the quality of the inversion methods, we use in addition a prediction problem. The prediction data will not be used by the inversion algorithms. Here we consider, the prediction of the 10-day groundwater protection zone, it is also referred to as 10-day capture zone (van Leeuwen et al. 1998). It is calculated in a slightly different setup but with the same geological model and its properties. The boundary conditions are prescribed heads equal to 1 m on the left boundary, 0 m on the right boundary and interpolated between those two values on the upper and the lower boundary. A constant pumping rate of $0.04\,m^3/s$ is imposed at the well and forward particle tracing is performed on the steady state solution of groundwater flow. The 10-day zone contains each location (pixel), from where groundwater reaches the pumping well in less than 10 days.

Inversion set-up

We will perform the inversion three times for each method. Every time, we use a different training image. The baseline case uses the reference TI, the two other cases use a different one. In this way, the robustness of the inversion methods can be tested. The two other training images were generated with the TI generator tool (Maharaja 2008) in the Ar2GEMS software. Their size is

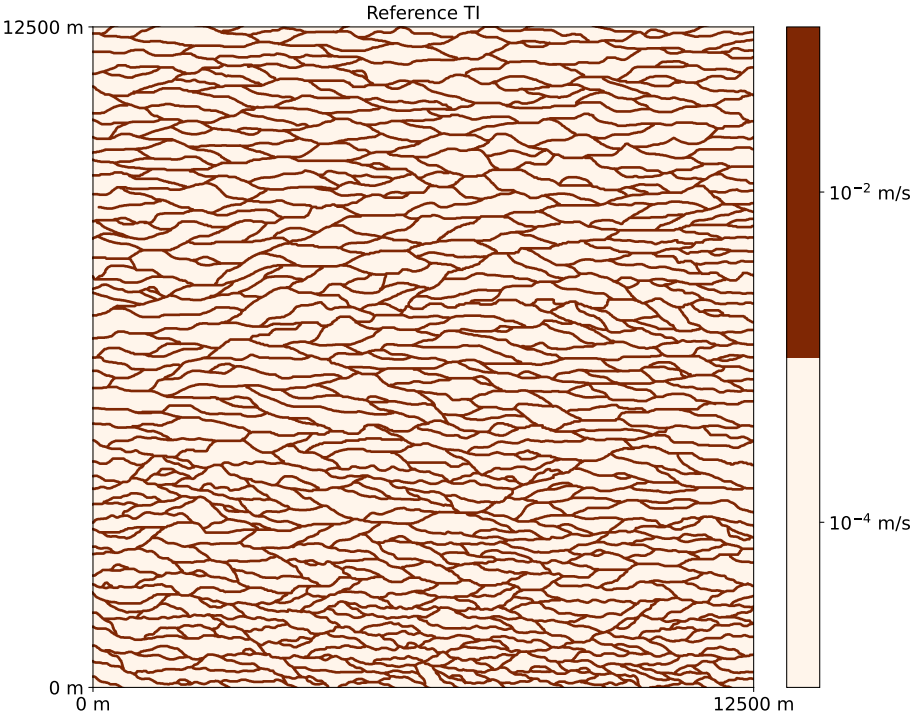

Figure 6.3: Reference (true) training image, used to generate the reference ("true") field. Data from (Zahner et al. 2016)

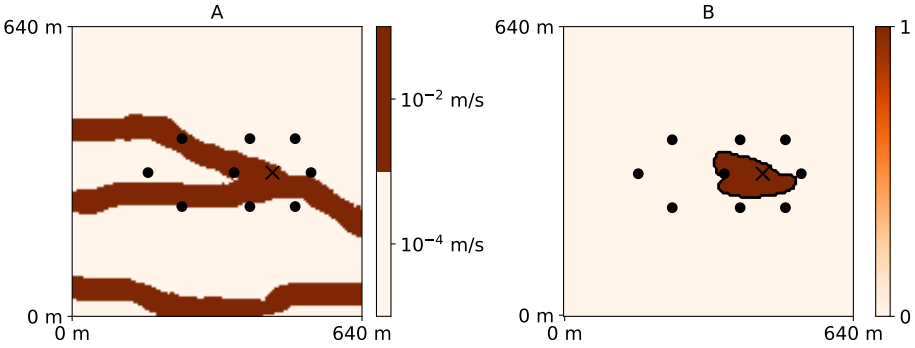

Figure 6.4: The reference field **(A)** used as the synthetic reality (considered unknown) and the corresponding 10-day groundwater protection zone **(B)**.

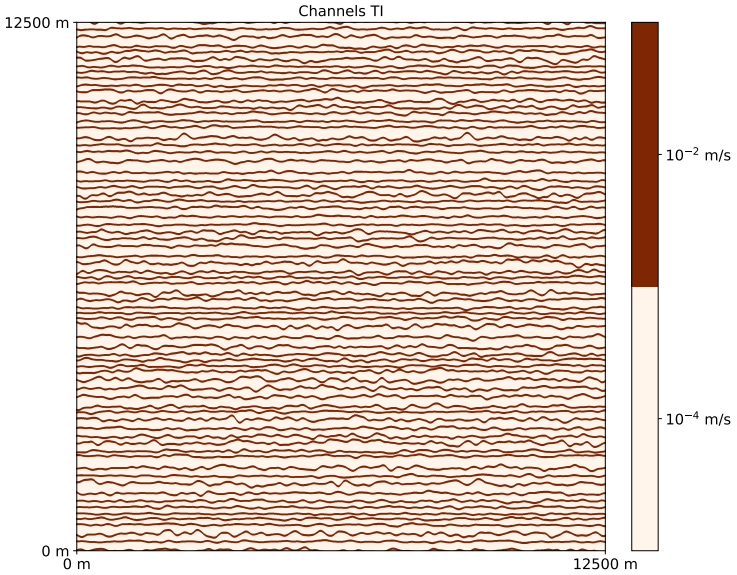

Figure 6.5: "Channels TI", an alternative training image.

identical to the one of the original TI. The first is a channelized medium with disjoint channels, and the second represents ellipsoidal deposits. TIs have the same proportion of facies as the original TI. We will refer to them as "Channels TI" (Fig. 6.5) and "Ellipses TI" (Fig. 6.6).

Both PoPEx and ESMDA benefit from the same geostatistical engine as the original reference; it means that the same DeeSse parameters are used in PoPEx + MPS and in ESMDA + MPS as were used to generate the reference. In theory, it is possible (but highly unlikely) that PoPEx samples the same model as the reference. It might not be the case for ESMDA, as it imposes dense conditioning on a coarse grid, but very similar models can be produced. The fact that PoPEx and ESMDA use DeeSse, gives them an edge compared to DREAM-ZS+GAN, as the reference is a realization generated by DeeSse (therefore not present in the TI). GAN is trained on images cut from the TI and learns to develop similar realizations, but it does not have access to samples simulated with MPS.

PoPEx was run with the following parameters: 32 parallel processes, and for a total of 50 000 iterations (50 000 forward runs), the maximal number of conditioning points is 10. ESMDA was run for 16 iterations (16 data assimilations), the size of ensemble is 128 and number of parallel processes 32 (2 048 forward runs). DREAM-ZS was run for total $T = 5\,000$ iterations with 32 parallel chains (160 000 forward runs). The initial archive size is set to 500. The size of the latent vector is $d = 50$. Frequency of snooker update is 0.1, $\delta_{\max} = 1$,

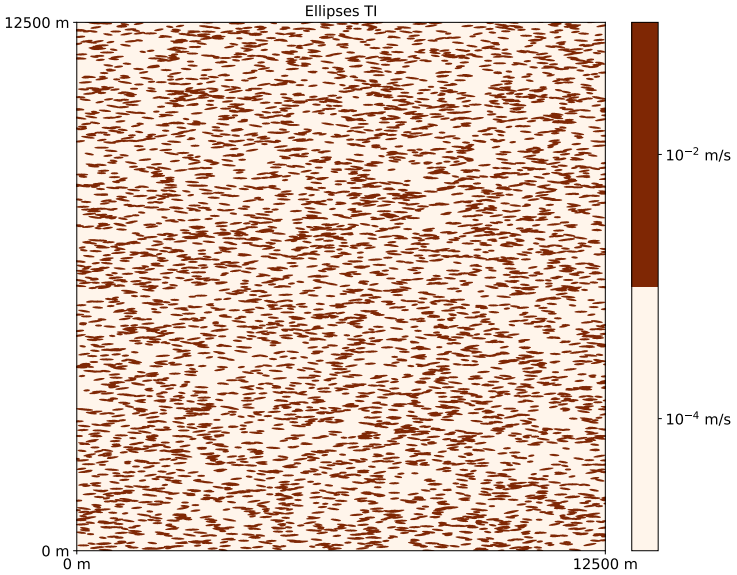

Figure 6.6: "Ellipses TI", an alternative training image.

and $n_{CR} = 3$, is according to recommendations for standard parameters. To study the convergence, we set a fairly low value for K, equal 10, but the value 100 should be sufficient for the chosen number of iterations. Moreover, we set $b = 0.05$ and $b^\star = 1 \times 10^{-6}$.

6.4 Comparing the results

Since our numerical experiments involve a large number of results, we need to use some summary statistics to compare the different methods efficiently. For that purpose, we will use three metrics for comparing the quality of the results:

1. The first metric indicates how well the measurements are reproduced by the simulation ensemble. Each continuous measured value will be compared with the ensemble of simulated values. A score which allows for such a comparison is the continuous ranked probability score (CRPS) used to assess probabilistic forecasts (Gneiting et al. 2007; Gneiting and Raftery 2007).

2. The second metric indicates how well the protection zone is predicted. In this case, a discrete (categorical) value is compared with a probability for each point using quadratic score (Gneiting et al. 2007).

3. Finally, the third metric indicates how well the geology is identified. For that purpose, we use the average of pixel-by-pixel quadratic score of predicted facies.

Below, we remind to the reader the definition of the CRPS and quadratic (Brier) scores.

CRPS score

The continuous ranked probability score (CRPS) is given by:

$$crps(F, x) = \int_{-\infty}^{\infty} \left(F(y) - \mathbb{1}(y \geq x)\right)^2 dy, \tag{6.39}$$

where F is the predictive cumulative distribution function (CDF), and x the true (observed) value, and $\mathbb{1}(y \geq x) = 1$ if $y \geq x$ and 0 otherwise. The advantage of the CRPS score is that it is expressed in the same units as the observations, and it generalizes absolute error, as when the CDF becomes a point observation, it equals the absolute error. In this way, it also provides a way to compare probabilistic and deterministic forecasts.

In our context, the *crps* score will be averaged for all measurement points:

$$CRPS = \frac{1}{N} \sum_{i=1}^{N} crps(F_i, d_i^{\text{obs}}), \tag{6.40}$$

where F_i is the predicted CDF corresponding to the measurement point d_i^{obs}.

Quadratic (Brier) score

The quadratic score was first introduced by Brier (1950) to quantify forecasts of categorical variables expressed in terms of probabilities. The quadratic (aka Brier) scoring rule is given by:

$$bs(\mathbf{p}, i) = \sum_{j=1}^{M} (\delta_{ij} - p_j)^2, \tag{6.41}$$

where $\mathbf{p}$ is a discrete probability distribution, e.g. $\sum_{i=1}^{M} p_k = 1$, and $p_k \geq 0$ for all $k \in \{1, 2, \ldots, M\}$, M the total number of categories, in this case equal to 2. δ_{ij} is the Kronecker delta $\delta_{ij} = 1$ if $i = j$, and 0 otherwise,

In our context, we will average the *Brier* score for all locations for which it is predicted if the location is in groundwater protection zone. Let $\mathbf{p}_k$ be a probabilistic forecast if point k is in groundwater protection zone, and l_k true category (in this case binary) of point k. the average Brier score is now given by:

$$BS = \frac{1}{N_l} \sum_{k=1}^{N_l} bs(\mathbf{p}_k, l_k), \tag{6.42}$$

with N_l the total number of locations. In the case of geology, the score is calculated in the same manner, but the facies data is used instead of the protection zone.

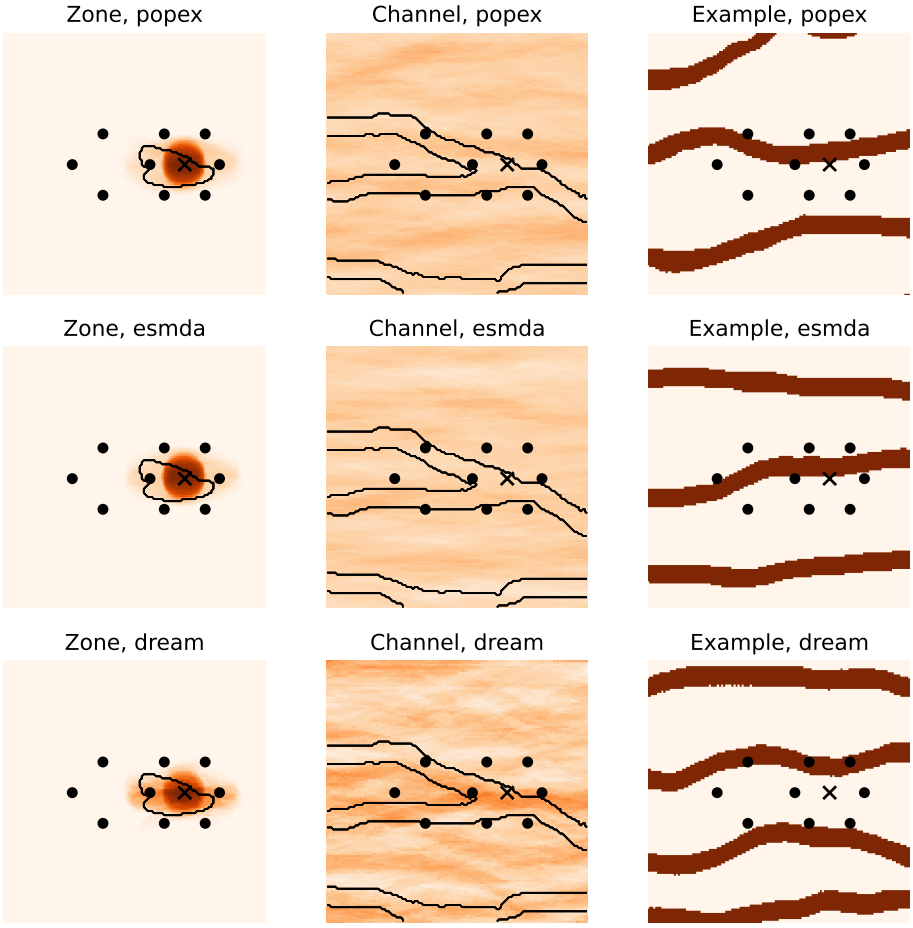

Figure 6.7: Prior probability distributions obtained with the Channels TI for the groundwater protection zone (left column), and the channel occurrence (middle column). The black contours correspond to the reference. The right column shows one example realization. The top row shows results obtained with PoPEx, the middle row shows results obtained with ESMDA, and the bottom row shows results obtained with DREAM-ZS. The colormap is the same as for Fig. 6.4**B**.

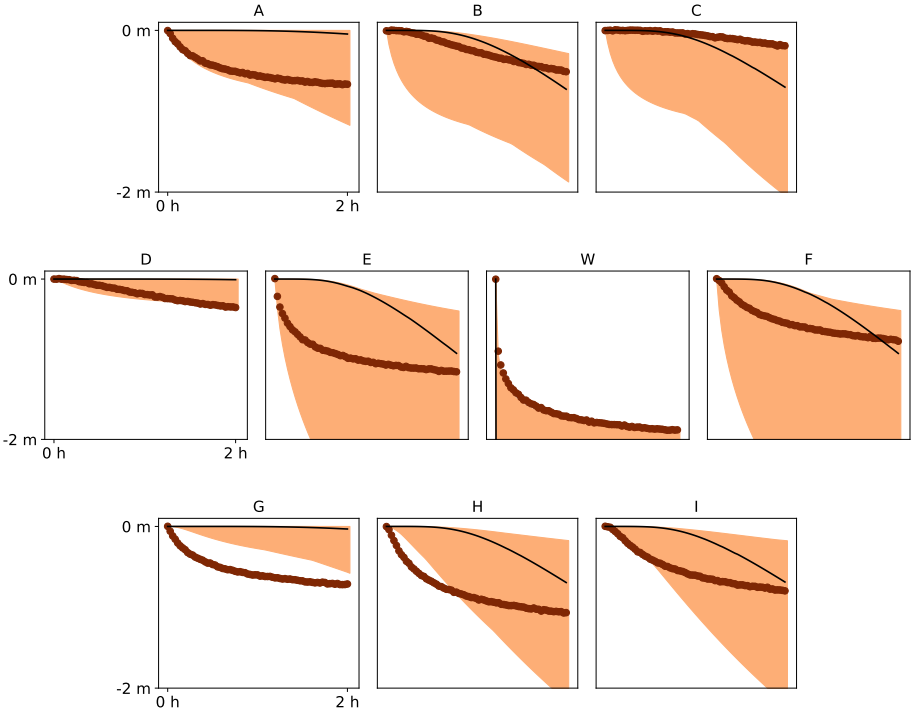

Figure 6.8: Prior distributions computed with DREAM-ZS and Ellipses TI for the time series of hydraulic heads recorded at the pumping well (W) and the nine piezometers (A-I). The observed data are marked with thick brown dots, the median of the prior distribution with thin black line and the 95 % confidence intervals are shown as shaded region.

6.5 Results

For each of the methods and for each of the TIs, we report prior and posterior data match, and corresponding prior and posterior probability maps (groundwater protection zone, facies) with examples of realizations. We also compare convergence of the method with respect to the number of forward calls. In this section, we do present one example of prior and posterior data match each; an example of the prior probability maps and all the posterior probability maps. The remaining results can be consulted in Appendix A.

Prior distributions

The prior groundwater protection zones are essentially circular (Fig. 6.7), as the most probable event is that the well is placed in the less impermeable facies. Such a configuration results in a large (and unrealistic) drawdown. If we look more closely at the prior groundwater protection zone maps, we

can see a second mode, which is an ellipse. This occurs when the pumping well intersects a channel. Such a zone shape can be visible for the true and channels TIs (Figs. A.19 and A.20), and less evident for the ellipses TI (Fig. A.21).

The channel prior probability maps are roughly homogeneous for all the simulation methods and training images (for example, see Fig. 6.7). This is expected as we did not impose any prior conditioning data, the prior probability of a channel corresponds then to the proportion of channels in the training image.

For the time series of head values in the piezometers and pumping well, the prior distribution shows wide confidence intervals, as can be seen for DREAM-ZS and Ellipses TI for example (Fig. 6.8). The true data are usually within the 95% confidence intervals, but there are some exceptions, and it happens that some head data lie outside this range.

Posterior distributions

Here, we summarize the main characteristics of the posterior distributions. The confidence intervals for the hydraulic heads are very much reduced, and they mostly match the data well (Fig. 6.9). For the posterior distribution computed using the reference TI, all the methods achieved a satisfactory data fit, with DREAM-ZS and PoPEx producing very narrow confidence intervals and an excellent match (Figs. A.12 and A.10). ESDMA produced wider confidence intervals, and did not reproduce closely piezometer data H (Fig. A.11). For the Channels TI, PoPEx produced visibly wider confidence intervals (Fig. A.13), and some piezometric data (A, F, H, I) are not matched perfectly, but the fit is still reasonable. ESMDA achieves slightly worse piezometer data fits (Fig. A.14), but, surprisingly, the well data is reproduced poorly, still giving high probability to the well placement in (or close to) less permeable region. DREAM-ZS achieves a very close fit and provides narrow confidence intervals (Fig. A.15). The Ellipses TI is the most difficult one. PoPEx does not match the data well for some piezometers (A,C,F) and produces quite wide confidence intervals (Fig. A.16). ESMDA produced again wide confidence intervals, and the heads data in the pumping well are again poorly represented (Fig. A.17). DREAM produced very narrow confidence intervals, but the data are sometimes not very well matched (Fig. 6.9F).

In terms of posterior probability maps, all methods solved reasonably well the inverse problem in the reference case. The protection zones probability maps are very close to the reference protection zone for all methods (Fig. 6.10). The permeable facies probability maps were able to represent the bifurcation. We note however some general trend to predict with over-confidence certain geological features. For PoPEx, on the right side of the map, the channel goes straight with a high probability, while in the reference it goes slightly to the bottom. For ESMDA algorithm, the posterior probability map suggests some "eye" feature to the left of the bifurcation. This pattern is not suggested by PoPEx. For DREAM-ZS, the "eye" structure is even more pronounced and the same map indicates with high probability a channel at the bottom, which only partially coincides with the channel in the reference realization. These rather high posterior probabilities of presence of certain geological features that are not present in the reference seem to correspond to some artifacts of the methods and not to features suggested by the statistics of the TI.

The case of the Channels TI (Fig. 6.11) posed more challenges for the inversion algorithms. The protection zone was only very well represented by the DREAM method. The zone obtained with PoPEx is more elongated and less influenced by the bifurcation. Indeed, PoPEx indicates a higher probability of channel only in the lower branch. Due to this TI with disjoint channels, it had difficulties in reproducing the branching of channels. ESMDA did not attribute a high probability of permeable facies near the pumping well. However, it managed to find the double channel on the left side of the field. DREAM performed

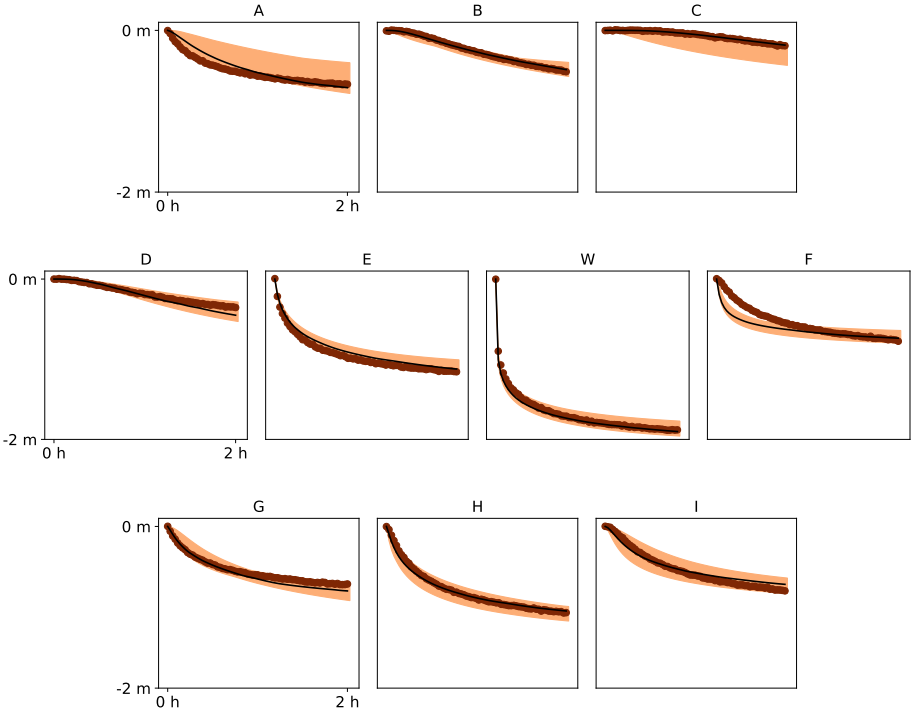

Figure 6.9: Posterior distributions obtained with DREAM-ZS and Ellipses TI for the time series of heads recorded at the nine piezometers (A-I) and pumping well (W). The observed data is marked with thick brown dots, the median of the distribution with thin black line and the 95 % confidence intervals are shown as shaded region.

best in reproducing the bifurcation, but it also displays channel artifacts in the upper part of the image.

The Ellipses TI can be considered as the hardest case, due to the disconnected nature of the high permeability features (the ellipses). The solution provided by PoPEx can be thought of as the most conservative, as it only places a blurred region of higher channel probabilities around the pumping well (Fig. 6.12). The advantage of this solution is the absence of high probability of channels in spurious zones. Nevertheless, the protection zone is not accurately reproduced. ESMDA placed a smaller protection zone than it should be and indicated with a high certainty the presence of channels in regions where they should not be. DREAM provides the most contrasted probability maps and placed incorrect geological features in the whole area.

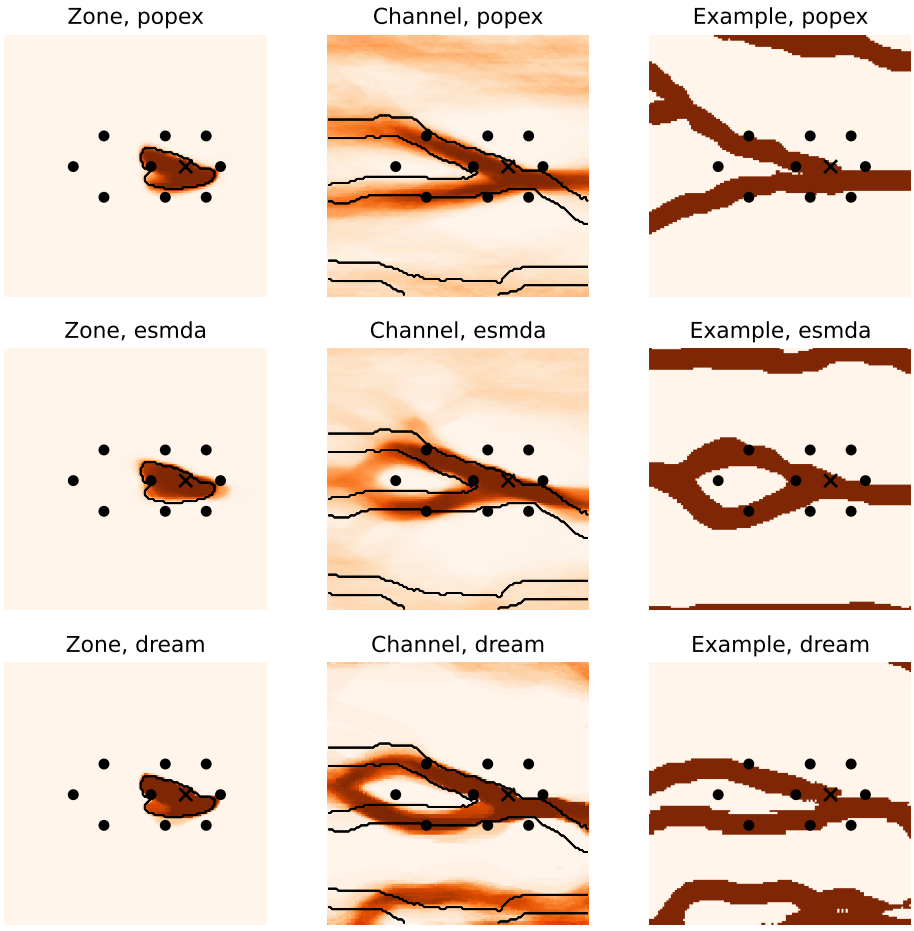

Figure 6.10: Posterior probability maps obtained with the true TI for the groundwater protection zone (left column), and the channel occurrence (middle column). The black contours correspond to the reference. The right column shows one example realization. The colormap is the same as for Fig. 6.4**B**.

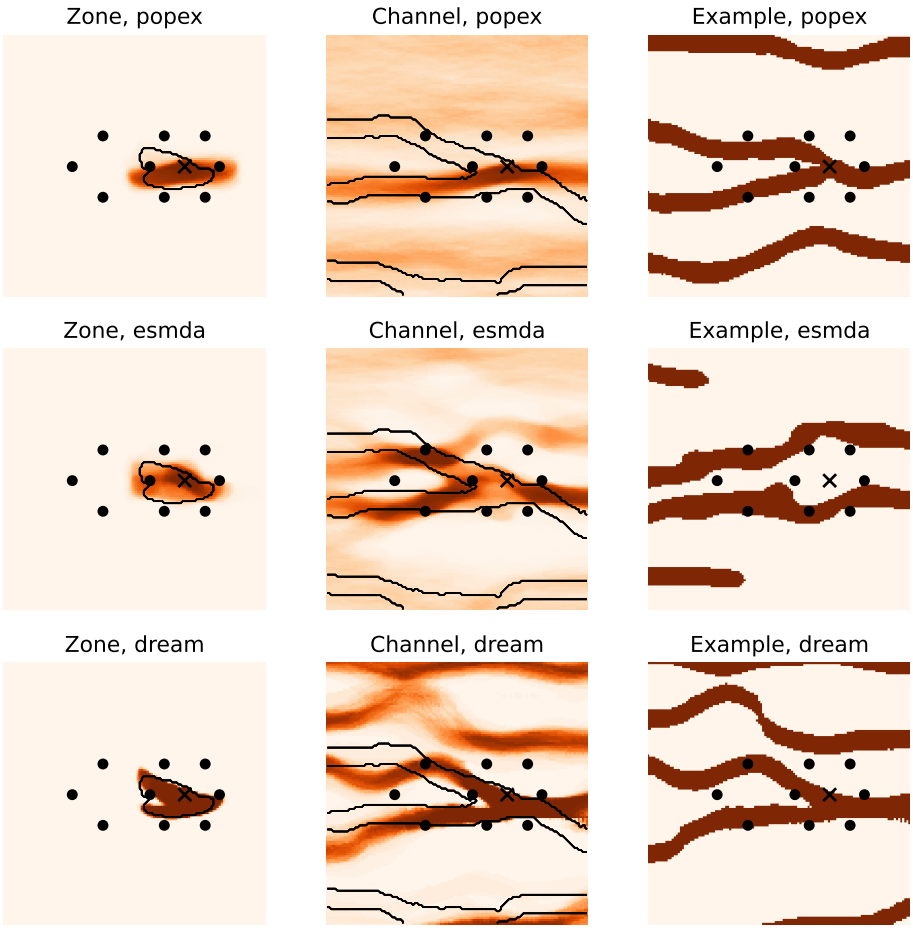

Figure 6.11: Posterior probability maps obtained and example realizations with the Channels TI.

Convergence and quality metrics

The previous comparison of the posterior distributions shows that it is not simple to compare visually the results. Therefore, in this section, we compare the three quality metrics convergence to get a better understanding of the performances of the methods. To facilitate the comparisons, we grouped the convergence plots by method in Figure 6.13, and by TI in Figure 6.14. The quality metrics are plotted as a function of the number of forward model runs, since these forward runs constitute the most expensive computational cost during the inversion.

Figure 6.13 shows that all the methods produced the best results (smaller values of the quality indicators) for the true training image. The methods also

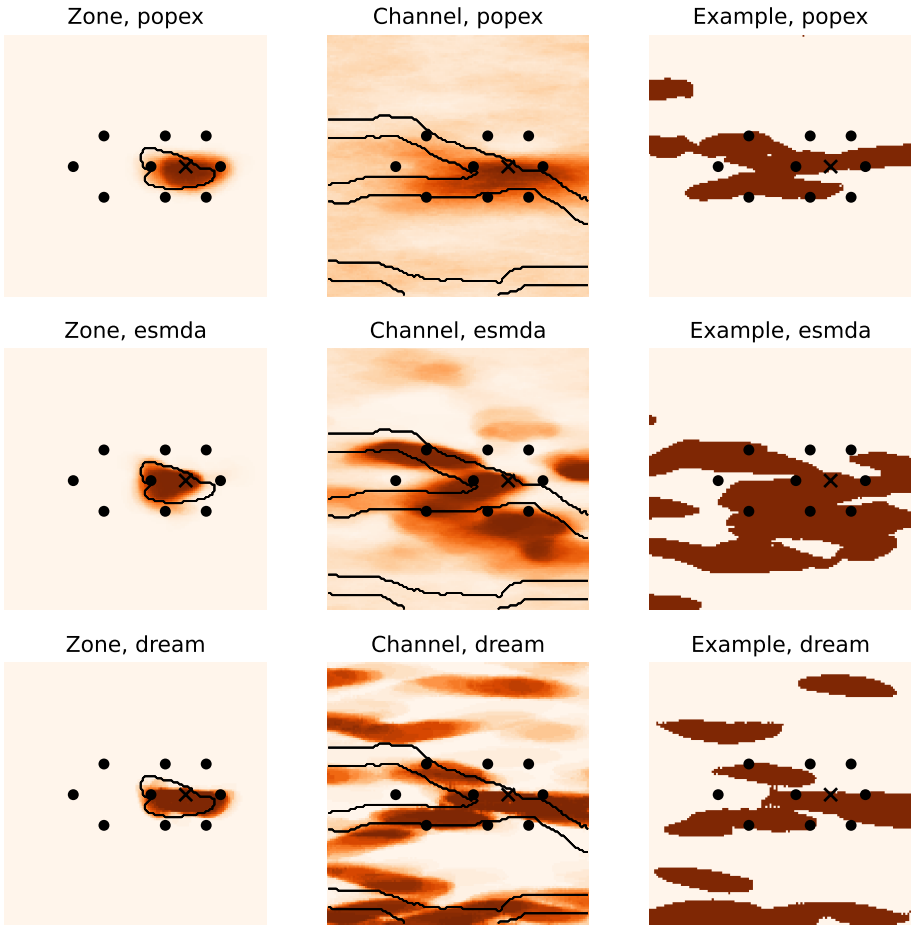

Figure 6.12: Posterior probability maps and example realizations obtained with the Ellipses TI.

converge, i.e. the errors diminish, with the number of forward calls (iterations) for the true training image. Generally, the scores are similar or better for the channel TI than for the Ellipses TI. While different scores show similar trends for the same case, a good CRPS score does not necessarily imply a good brier score. The most notable example of the lack of correlation is the case of DREAM algorithm. The CRPS score goes down for all TIs rather fast, and the scores for channels TI and ellipses TI are close. However, when comparing the BS-zone scores, the ellipses TI results become significantly worse than those with the channels TI. Moreover, the BS-channel score increases after 1×10^4 forward calls. It seems that the realizations collapse on a very similar (and not fully correct) model realization, and produce geological artifacts which are highlighted by this score, while the CRPS score on data match remains low.

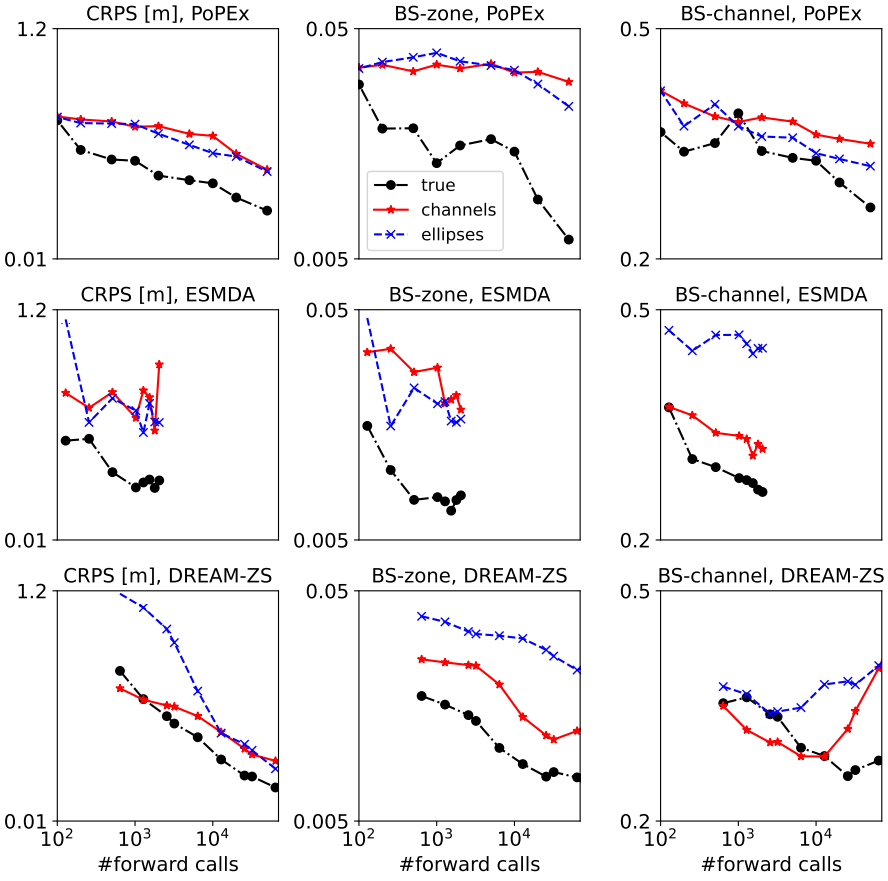

Figure 6.13: Convergence of the posterior predictions, as measured by different scores (columns) for every method (rows). Each plot compares the curves for different TIs.

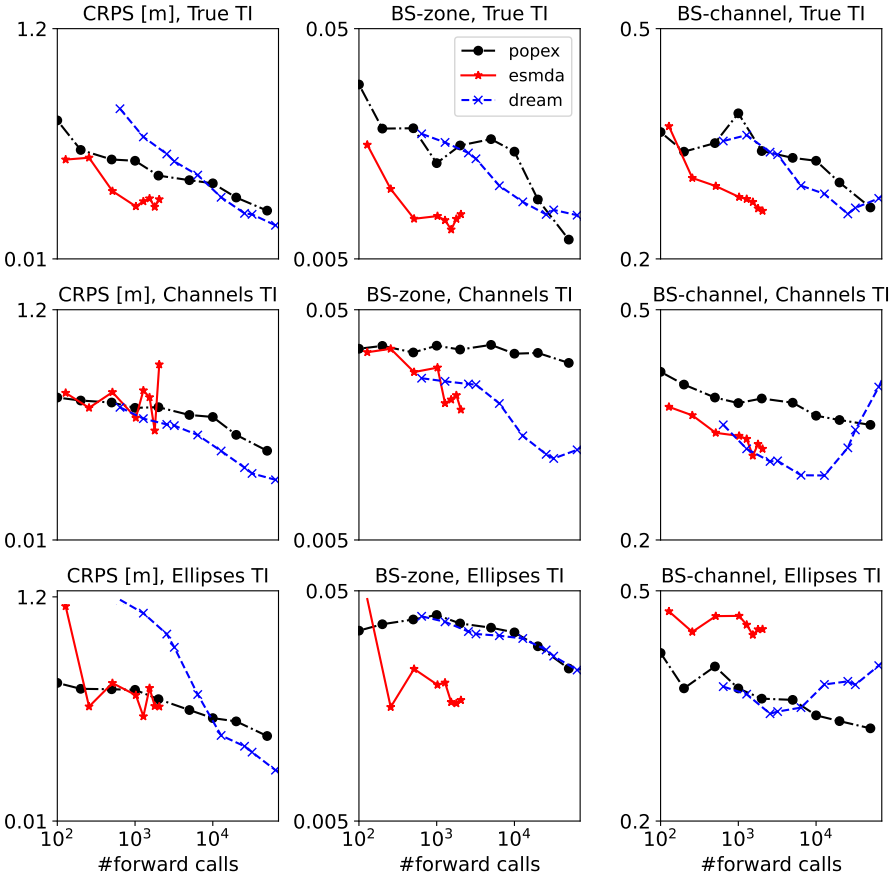

Figure 6.14: Convergence of posterior predictions, as measured by different scores (columns) for every TI (rows). Each plot compares curves for different methods.

When comparing the different methods for the same TIs (6.14), we note that ESMDA exhibits a very fast convergence at the beginning and then reaches a plateau. The method stops after relatively few forward runs (as compared to the other methods), and this is related to the choice of the parameter governing the number of data assimilations. Here, the default value of 16 is used (Lam et al. 2020) and it based on recommendations of Emerick and Reynolds (2013). In a sense, it is the least computationally expensive method, but the achievable quality is limited. Other methods can provide results of better quality, but they need more iterations. DREAM-ZS shows the best convergence for the data match and protection zone reproduction for the channels TI, but the error diverges for BS-channel and becomes larger than those of PoPEx and ESMDA for the largest number of iterations. The last row of Fig. 6.14 shows a slow but steady convergence of PoPEx when using the Ellipses-TI, as opposed to ESMDA, which stagnates after the first iterations, and DREAM-ZS, which matches the piezometric data very well but has slow convergence on the BS-zone and even diverges on the estimation of the geological features (BS-channel).

In summary, PoPEX is the method that seems to be the most robust. It converges steadily with the number of forward model calls, and rather fast when the proper training image is given but slowly if a wrong training image is provided. On the opposite, ESMDA always improves the solution very fast for the first iterations even with a wrong training image, but it stagnates rapidly after a few iterations. DREAM has an intermediate behaviour, it can be faster than PoPEx and continues to improve the results when ESMDA stagnates even with a wrong TI. But there are cases where the method diverges and the error increases with additional iterations.

6.6 Conclusions

In this study, we compared three recent stochastic inversion methods capable of working with categorical fields: Posterior Population Expansion (PoPEx) with multiple-point statistics (MPS), Ensemble smoother with multiple data assimilation (ESMDA) with MPS pyramids, and DREAM-ZS with Wasserstein generative adversarial network (WGAN). A synthetic test case with hydrogeological data (time series of hydraulic heads) was used, and two geological facies were considered in this inverse problem. The results were analyzed both for the inversion and for a prediction of the 10-day groundwater protection zone. The quality indicators took into account the reference solution (ground truth represented by the reference solutions) with probability forecasts.

Our main finding is that when the methods were given the correct prior information (represented by a training image), they all achieved reasonable convergence. Even with wrong priors, some acceptable solutions were obtained. However, the choice of the prior is essential. The convergence was negatively affected when the TI with lenticular deposits was used. The TI with disjoint channels (as opposed to the original TI with bifurcating channels) provided

slightly better results than ellipsoidal deposits, but the two wrong TIs introduced unwanted effects.

We observed that none of the methods performed systematically better than the others.

The advantage of PoPEx is that it presented the most steady convergence compared with ESMDA and DREAM-ZS, whose scores fluctuated (ESMDA) or even increased (DREAM-ZS) with the number of iterations. PoPEx can also be thought of as the most "conservative", as it does not introduce "artifacts", e.g. geological features which are unlikely (outside the informed zone). However, this "conservative" approach leads to poorer data fits and worse predictions of the protection zone in certain cases. In the case of a wrong prior, it rarely performed better than the other methods.

The advantage of ESMDA is its fast convergence. It was able to reasonably identify the permeable zones between the piezometers even with wrong priors, producing patterns not present in the TI, which was needed to match the data. But it also indicated with high certainty some permeability patterns outside this zone that were incorrect. A surprising point and drawback of the method is that it did not always place the well in the permeable zone when a wrong prior was used. It resulted in overestimated confidence interval for the well data.

DREAM-ZS often matched the data the best and provided very good protection zone estimates. It was able to generate realizations which had bifurcation patterns without seeing them in the training set. Outside the informed zone, it was often over-confident in placing geological patterns, even to a greater extent than ESMDA. However, the good performance of DREAM-ZS is remarkable, as the reference data was generated using the MPS tool employed by PoPEx and ESMDA. Identifying these geometries should be easier for these methods than for DREAM. For a fairer comparison, an additional reference realization generated with GAN could be added, but it would require repeating the whole study.

Further extensions of this study might include solving the inverse problem with ESMDA+GAN (Canchumuni 2021; Li and Redoloza 2020), which would be interesting to compare especially with ESMDA+MPS and DREAM-ZS+GAN. Another point is the missing probabilistic reference solution, as for example showed by Jäggli et al. (2017). However, generating such a probabilistic reference solution is computationally very expensive in the cases, where a lot of observations are available. Moreover, in this case, a separate probabilistic reference solution would be desired for every prior. An interesting option would be a multi-categorical reference. PoPEx is ready to handle such a case and adapting GAN should be staightforward, but ESMDA+MPS must be adapted accordingly to handle Gaussian pyramids for every category.

Chapter 7

Conclusion and perspectives

In this thesis, we have developed a few tools and frameworks for facilitating the discrete stochastic inversion. Two key point were addressed: the choice of the prior and the computational cost. In the last chapter we have seen that the choice of the prior is essential and affects the convergence of the inverse method. A comparison study of different novel inversion methods was presented with a framework for comparing the quality of the results.

7.1 Main contributions

The main contributions presented in this thesis are divided in three parts: the first concerns the selection of the prior (cross-validation framework and Direct Sampling Best Candidate algorithm); the second concerns the acceleration and convergence of the inversion algorithms (application of machine learning to speed up population expansion (PoPEx algorithm), the tempered likelihood for PoPEx, and the third comparison of inversion methods).

Importance of prior in stochastic inversion

The choice of the geostatistical simulation method and its parameters is crucial in many contexts. In Chapter 6, we have seen a prominent example, where the choice of the prior (represented by the selection of one parameter of multiple-point statistics geostatistical method — the training image) played the key role in convergence of the inversion. In that categorical example, prior conditioning data was not available, but if it were given, the cross-validation framework could be used to help select the best prior. The proposed generic cross-validation framework for categorical simulations provides a systematic way to assess quality and compare geostatistical algorithms and their parameters.

To further simplify the choice of algorithmic parameters in the context of multiple-point statistics, the Direct Sampling Best Candidate algorithm was introduced. It reduces the necessity of tuning three main parameters to two

with respect to the standard Direct Sampling algorithm, but retains all of its advantages. We used this algorithm with PoPEx and ESMDA in the comparative study of inversion frameworks.

Speeding up inversion

We showed that inversion can be accelerated using fast and easy to implement machine learning techniques. Using a tracer data inversion case, we found that ML algorithms can speed up PoPEx algorithm up to two times. The ML framework is generic and could be tested with other algorithms. The proposed speed-up score allows to take into account proportion of time spent on forward and easily estimate the actual speed-up without actually running inversion.

The tempered likelihood fixed the problem of weight degeneracy in PoPEx and therefore opened the possibility to apply it to cases with even more data; alleviating the need of reducing the dimensionality of the problem. Moreover, it simplified the generation of predictive weights. The flexibility of the tempered likelihood was demonstrated on the tracer test problem, and it was used in the final comparison of inversion methods.

Comparison of inversion methods

As the final piece of the thesis, I presented a synthetic inverse problem — a pumping test case with geology modelled with two facies. Three inversion methods were used to solve the problem and their performance was compared using continuous ranked probability score (CPRS) and quadratic (Brier) score. The three inversion algorithms were the following: PoPEX+MPS, ESMDA+MPS and DREAM-ZS+GAN. For the MPS, the DSBC algorithm was used, and tempered likelihood was applied in PoPEx. All the algorithms were able to solve the problem in the baseline case, but provided degraded results quality when the prior was not correct. The probabilistic scores allowed to compare the results with the single synthetic reference. We identified the shortcomings and advantages of the methods. In general, all methods performed well given the right prior. ESMDA has the fastest convergence and requires the least forward model runs, but the achievable quality is limited. DREAM-ZS was able to match data very well also in the cases with the wrong prior, but produced geological artifacts: was overconfident placing geological pattern, where they should not be identified. It was not the case of PoPEx, which appeared to be most robust in this matter, not updating geological probability maps without sufficient information, but its convergence was somewhat slow in some cases.

7.2 Perspectives

I suggest two main directions for further research. The first concerns the applications of the cross-validation framework, and the second development of methodologies for quality assessment of the inverse problem solutions.

Applications of cross-validation

The cross-validation framework was tested on two-dimensional examples, but it would be interesting to apply it on a three-dimensional case, and develop a methodology for treating borehole data, which are spatially arranged along vertical lines. Indeed, borehole data are not distributed randomly in depth; on the contrary, when some points in wells are missing, they can be reconstructed by the other. This problem is not only of practical importance, but also of theoretical interest, as a grouped K-fold approach should be probably chosen. Randomly removing points will not give a good estimation of prediction error. Therefore, grouping technique (by wells or parts of wells) might be a solution, but some research might be done to look for alternative strategies (e.g. when there are not many boreholes).

Quality indicators for inverse problem

Robust and universal scores are desirable, as they would encourage more comparisons of inverse methods, and development of their performance. In the inversion comparison chapter, we have used CRPS and Brier scores for comparing inversion results. We were able to use them, as the reference geology or capture zone was available. In real scenario, the reference solution is unknown, and only CRPS score can be computed, but quality indicators are still needed to estimate if the convergence has been achieved.

A possible solution could be the application of cross-validation to the observation data, similarly to what we have seen in the geostatistical case. It would require re-running inversion with fewer data several times (for example, 5 times in 5-fold CV). This approach would therefore be computationally expensive, but cross-validated CRPS score might be more robust than the standard CRPS score. The cross-validation technique should be developed carefully, as measurements are often correlated, and therefore the splitting scheme must be appropriately adapted.

The comparisons of the inversion methods could be extended to include additional cases. For example, an additional test set-up should include references generated by GAN and pyramid MPS for a fairer comparison among the methods. Combination of ESMDA and GAN is also a promising method, as suggested by literature.

Another interesting direction to accelerate the inversion, is the combination of adjoint method with GAN and Monte Carlo. Such type of approaches have already been investigated in context of a seismic forward problem (Mosser et al. 2019), and a hydrogeological set-up could follow this idea. This would require, however, using an adjoint of the groundwater flow or transport solver.

Finally, an adaptation of the ESMDA method is required to allow running multi-categorical inversion. A straightforward possibility is to use a single

Gaussian pyramid that agglomerates the information for all the facies in one continuous variable, but a better treatment of geology would require using one pyramid per facies (Straubhaar et al. 2020), and thus would require assimilating more parameters.

Appendix A

Supplementary material

A.1 DSBC

In this section, we provide some additional information and results related to the tests of the DSBC methods conducted in chapter 3.

x	y
1200	500
800	500
1700	600
2600	800
1900	900
2200	700
1000	900
500	800
2700	1200
1500	1000

Table A.1: Bottom left positions extracted images.

n	f	t	MPE	MVE	time (s)
8	0.005	0.002	1.76	0.64	1.2
8	0.005	0.010	1.77	0.65	1.1
8	0.005	0.005	1.80	0.67	1.1
32	0.005	0.001	1.70	0.67	2.9
16	0.005	0.005	1.77	0.68	1.2
16	0.005	0.001	1.72	0.68	1.8
8	0.005	0.001	1.79	0.69	1.3
64	0.005	0.001	1.75	0.72	5.8
16	0.005	0.010	1.84	0.73	1.1
16	0.005	0.002	1.78	0.74	1.5
32	0.005	0.002	1.79	0.77	2.1
32	0.005	0.005	1.90	0.81	1.4
64	0.005	0.002	1.85	0.82	3.8
32	0.005	0.010	1.96	0.83	1.1
64	0.005	0.005	2.04	0.97	1.9
64	0.005	0.010	2.12	1.01	1.1

Table A.2: All the tested DS parameter sets in Case 2 with corresponding scores sorted according to MVE score.

n	f	MPE	MVE	time (s)
16	0.0005	1.58	0.58	2.2
16	0.0002	1.59	0.58	2.2
16	0.0010	1.58	0.59	2.9
32	0.0010	1.61	0.61	3.2
8	0.0010	1.66	0.61	2.6
32	0.0005	1.62	0.63	2.2
8	0.0002	1.68	0.65	2.2
32	0.0002	1.65	0.65	2.1
8	0.0001	1.71	0.66	2.3
8	0.0005	1.69	0.66	2.0
16	0.0001	1.61	0.66	2.3
32	0.0001	1.68	0.67	2.3
64	0.0002	1.82	0.73	2.2
64	0.0010	1.76	0.73	4.1
64	0.0005	1.78	0.73	2.7
64	0.0001	1.88	0.78	2.5

Table A.3: All the tested DSBC parameter sets in Case 2 with corresponding scores sorted according to MVE score.

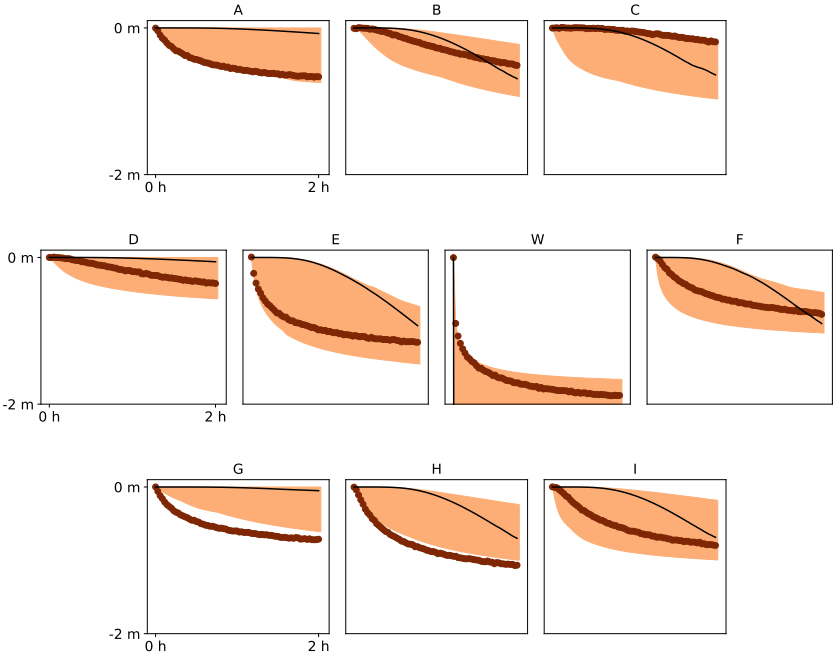

Figure A.1: PoPEx and true TI: prior distribution of heads.

A.2 Comparison of inverse methods

In this section, we provide additional results obtained during the comparison of the inverse methods in Chapter 6. In particular, we provide the complete set of figures illustrating systematically the results obtained with the different inverse methods.

Data match

We start by comparing the head data during the pumping test with the head computed from the prior ensemble and the posterior distribution. In all these figures, the observed data is marked with thick brown dots, the median of the ensemble is shown with a thin black line and the 95 % confidence intervals are shown as shaded region.

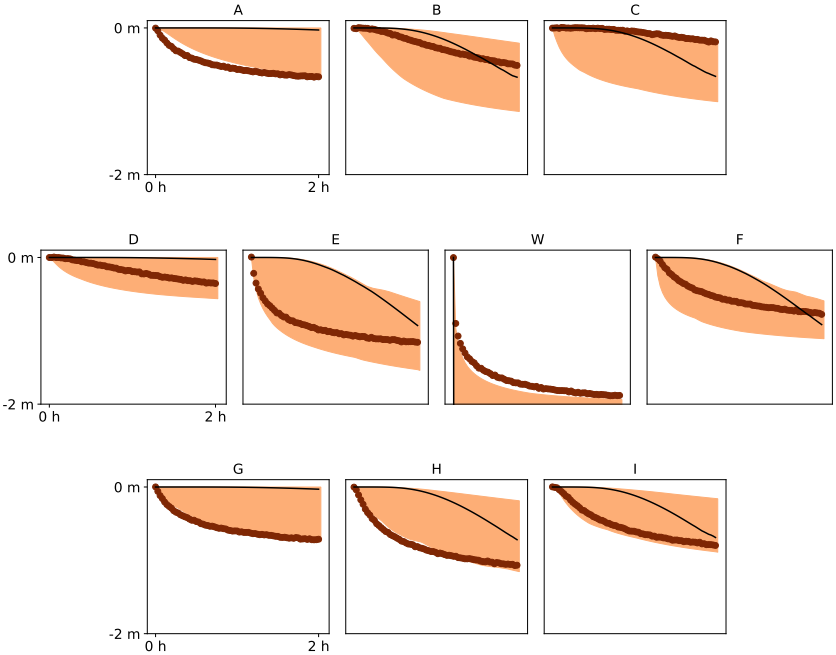

Figure A.2: ESMDA and true TI: prior distribution of heads.

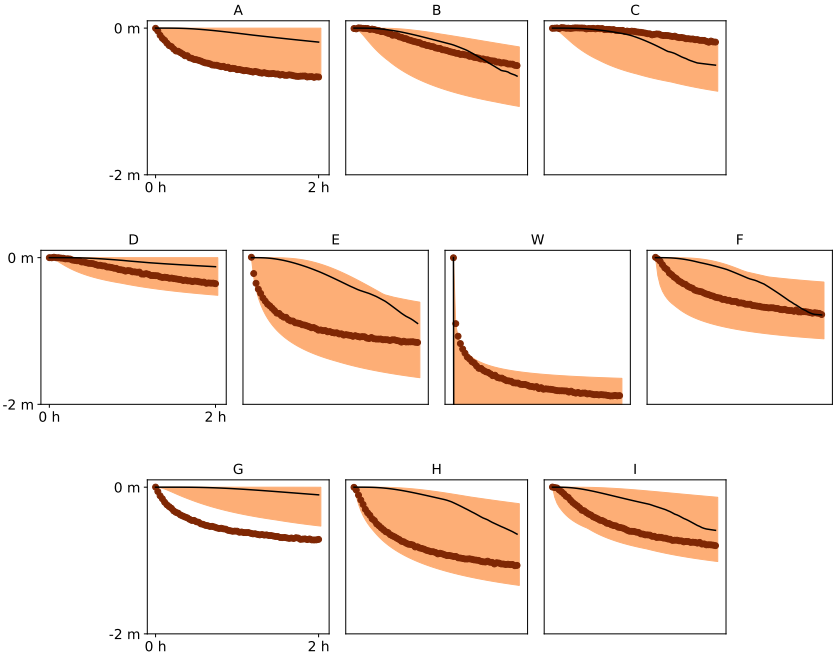

Figure A.3: DREAM-ZS and true TI: prior distribution of heads.

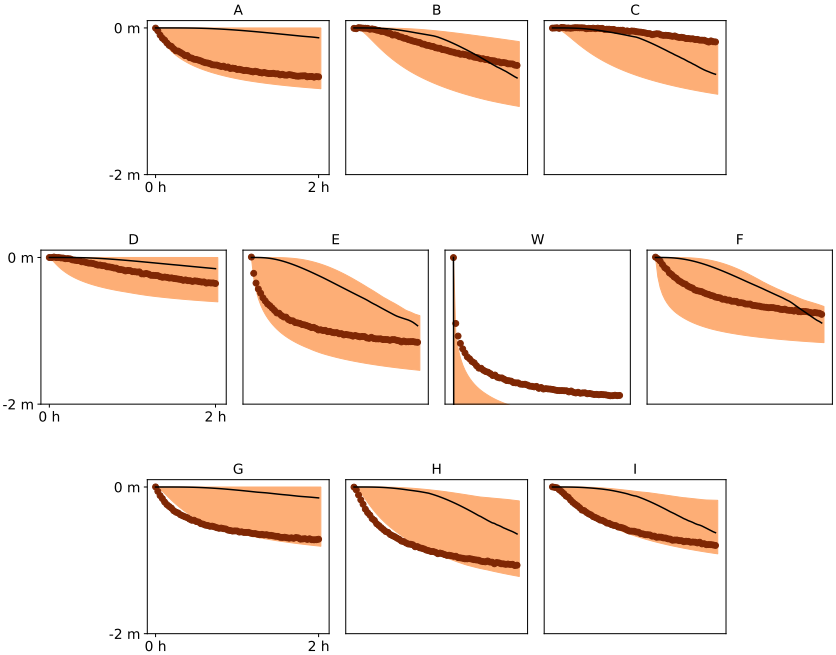

Figure A.4: PoPEx and Channels TI: prior distribution of heads.

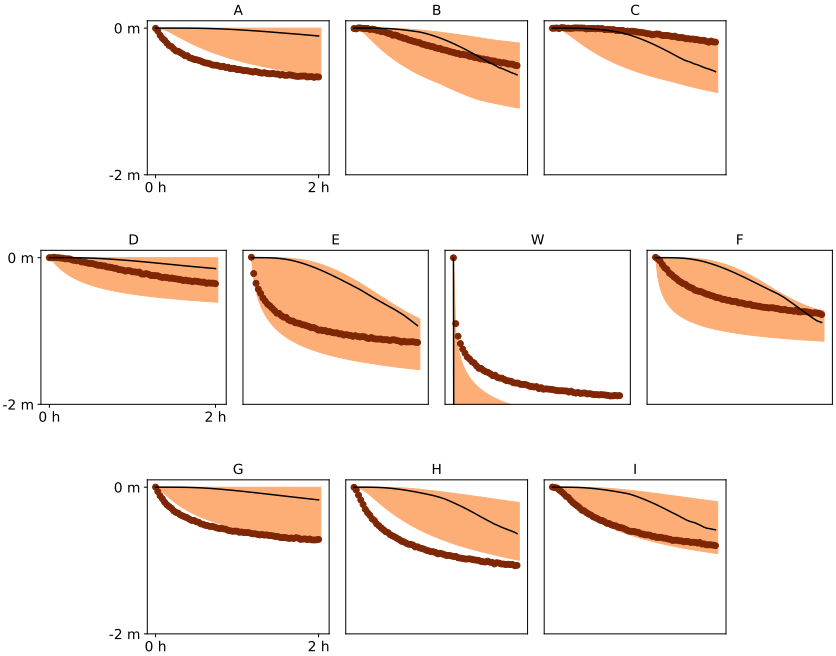

Figure A.5: ESMDA and Channels TI: prior distribution of heads.

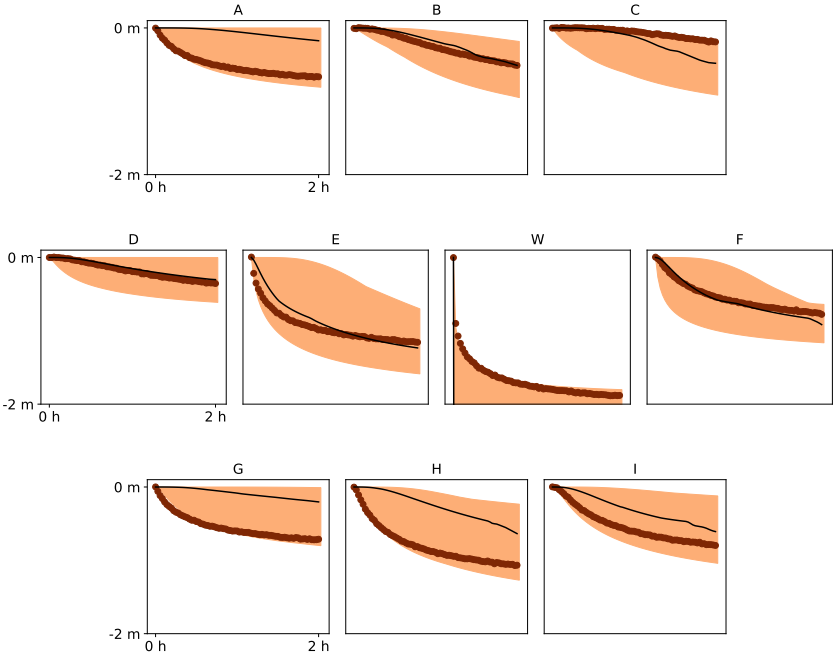

Figure A.6: DREAM-ZS and Channels TI: prior distribution of heads.

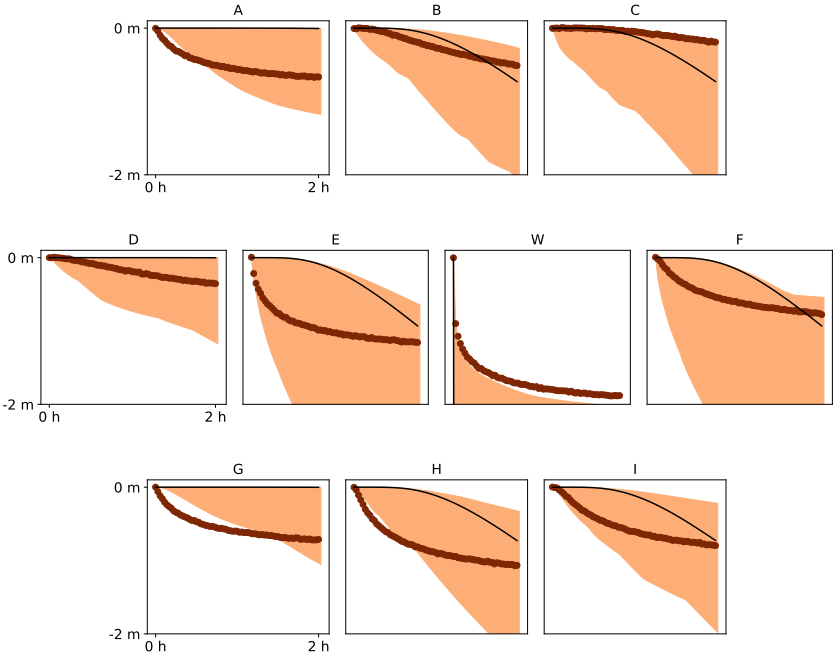

Figure A.7: PoPEx and Ellipses TI: prior distribution of heads.

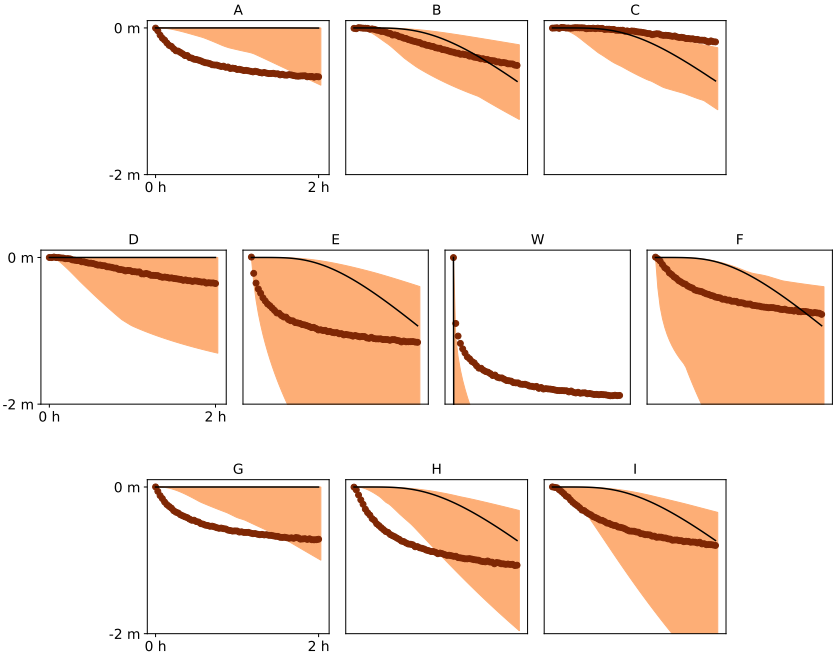

Figure A.8: ESMDA and Ellipses TI: prior distribution of heads.

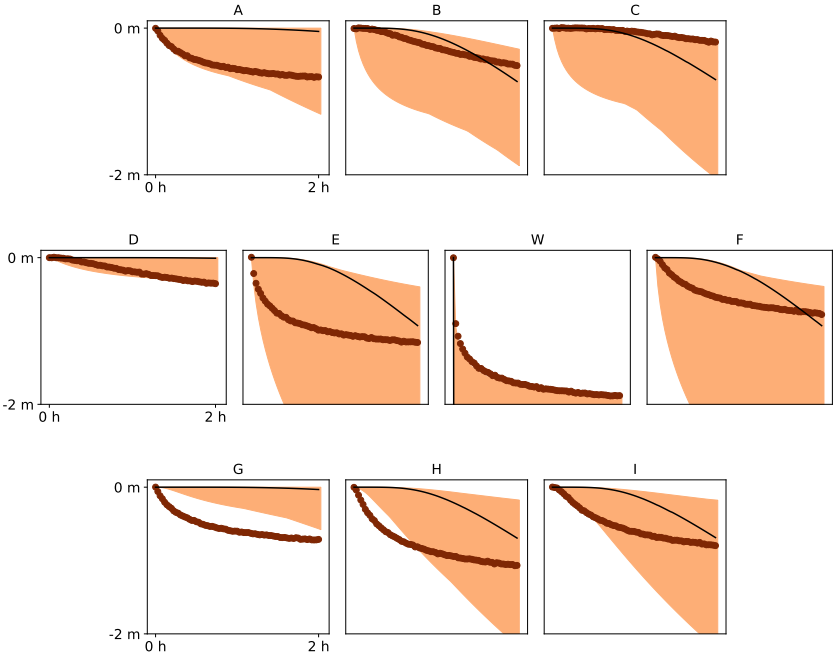

Figure A.9: DREAM-ZS and Ellipses TI: prior distribution of heads.

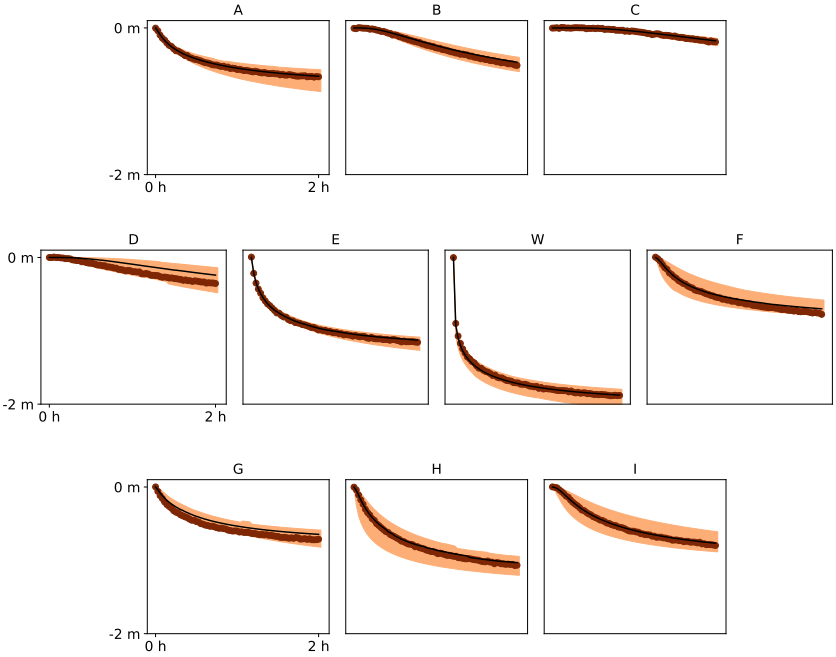

Figure A.10: PoPEx and true TI: posterior distribution of heads.

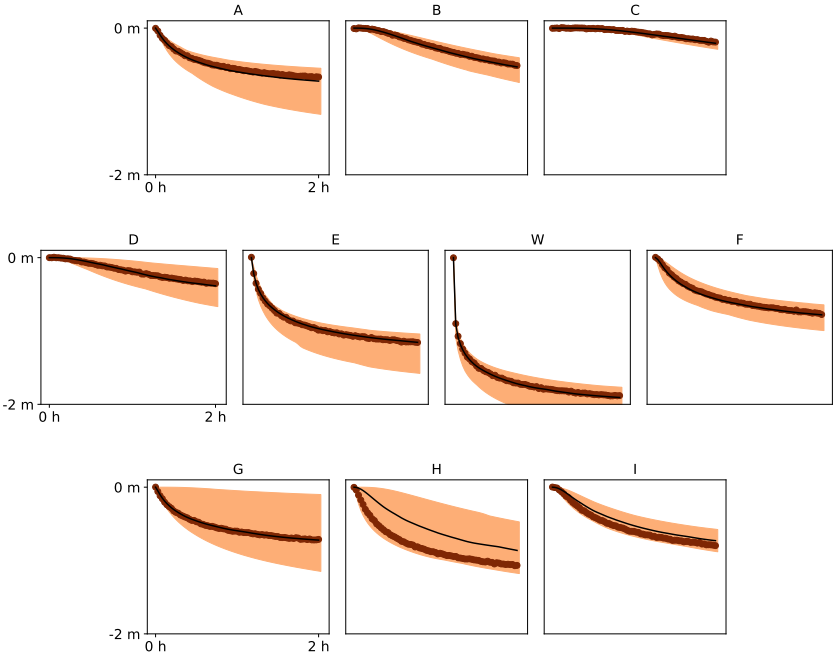

Figure A.11: ESMDA and true TI: posterior distribution of heads.

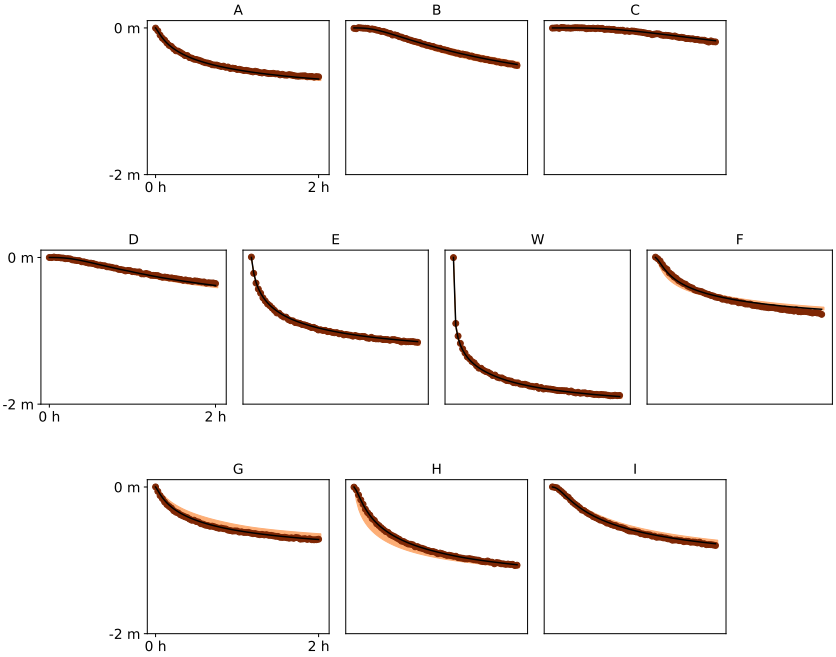

Figure A.12: DREAM-ZS and true TI: posterior distribution of heads.

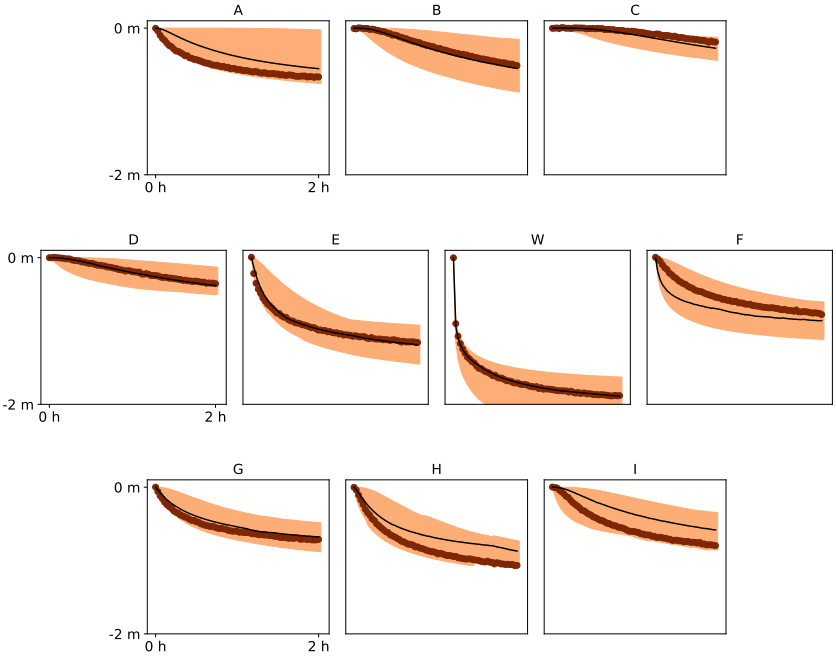

Figure A.13: PoPEx and Channels TI: posterior distribution of heads.

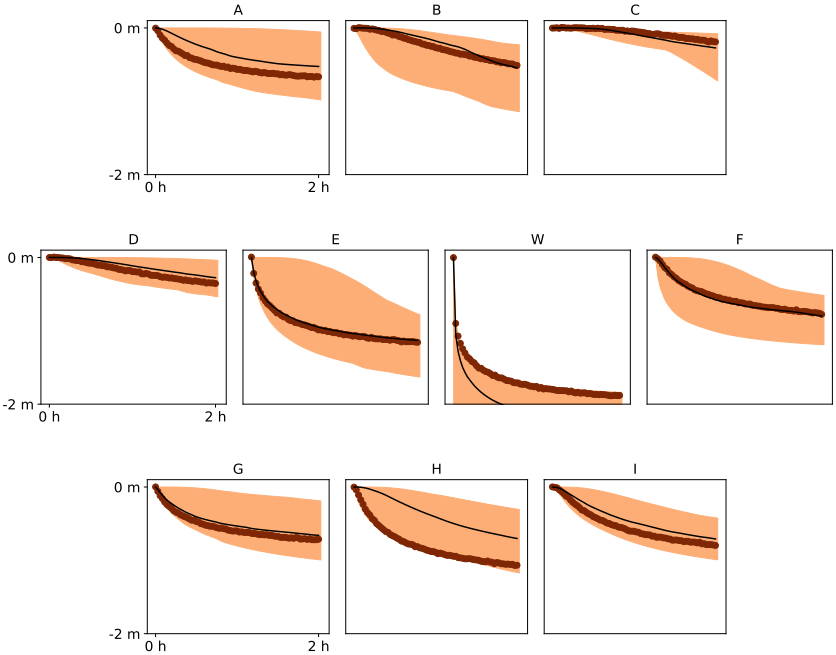

Figure A.14: ESMDA and Channels TI: posterior distribution of heads.

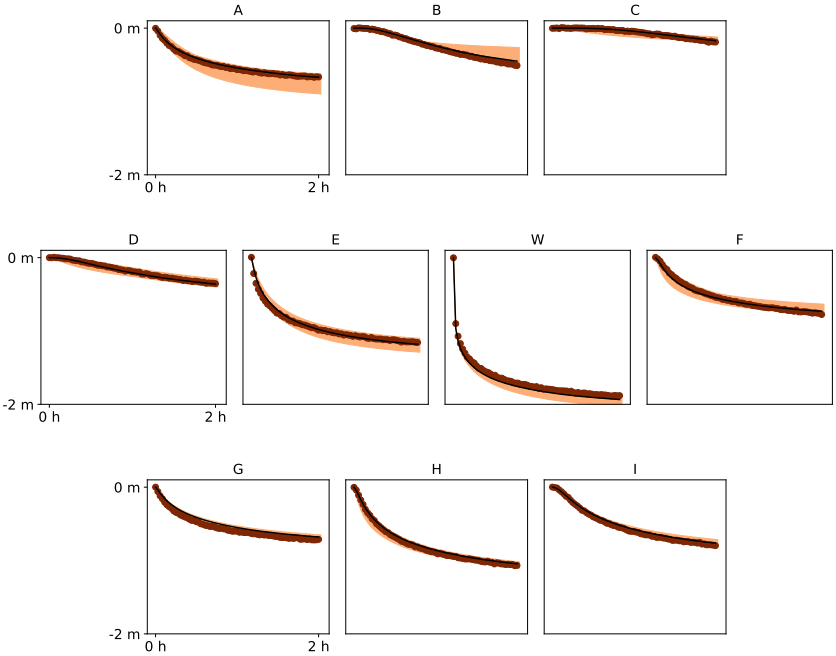

Figure A.15: DREAM-ZS and Channels TI: posterior distribution of heads.

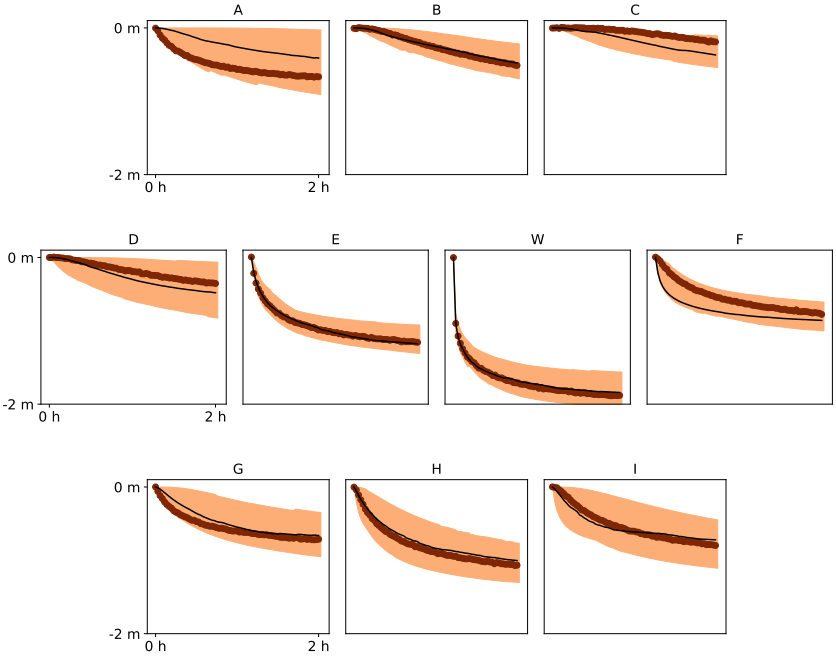

Figure A.16: PoPEx and Ellipses TI: posterior distribution of heads.

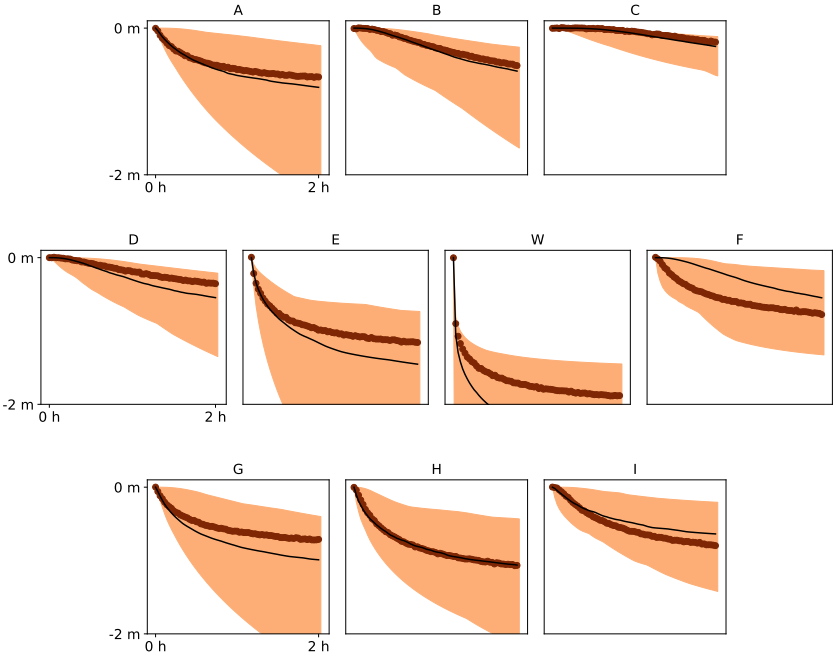

Figure A.17: ESMDA and Ellipses TI: posterior distribution of heads.

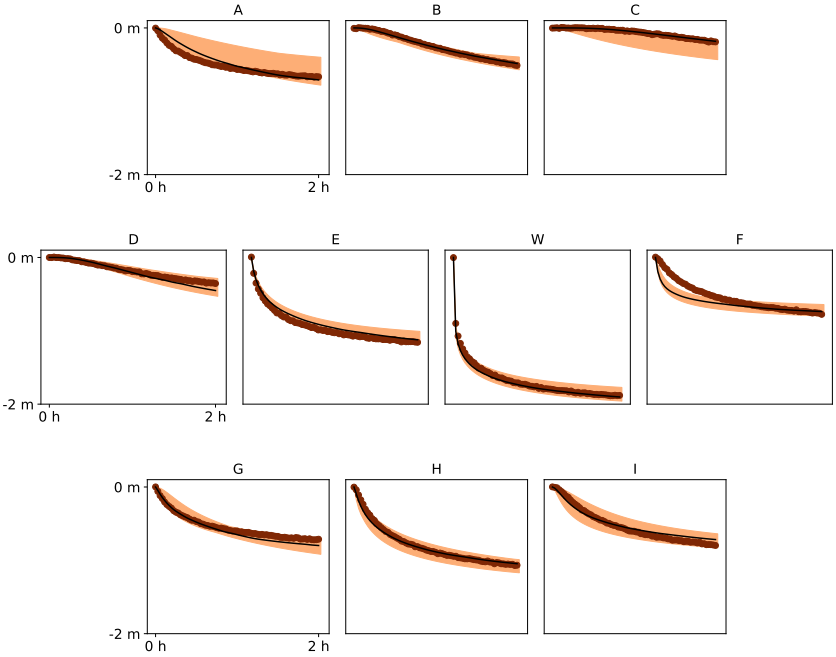

Figure A.18: DREAM-ZS and Ellipses TI: posterior distribution of heads.

Prior Predictions

In this section, we present the results of the prior predictions obtained with the three methods discussed in Chapter 6 in order to show the effect of the TI on the prior. All the figures have the same organization. The left column shows the 10 days groundwater protection zone. The middle column shows the channel (or high permeability facies) probability maps. The right column shows an example of one single realization. In each figure, the top row shows results obtained with PoPEx, the middle row results with ESMDA, and the bottom row results with DREAM-ZS. The black contours correspond to the reference. The colormap is the same as in Figure 6.4**B** on page 116.

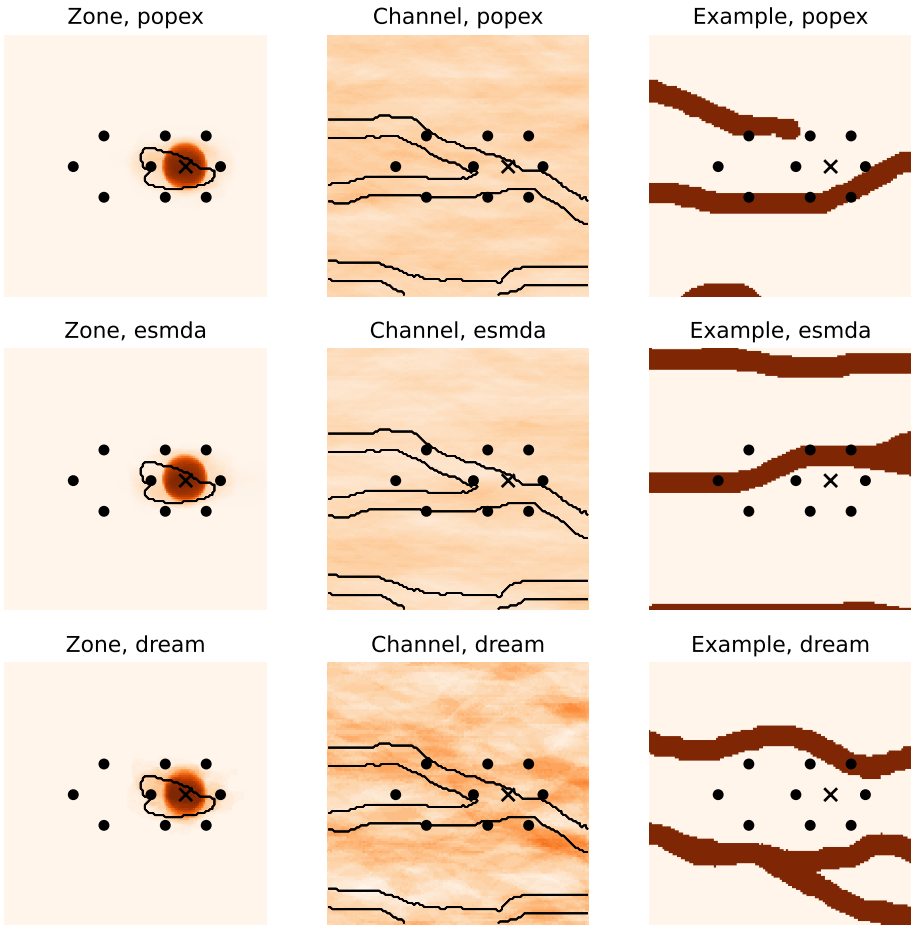

Figure A.19: Prior predictions with true TI.

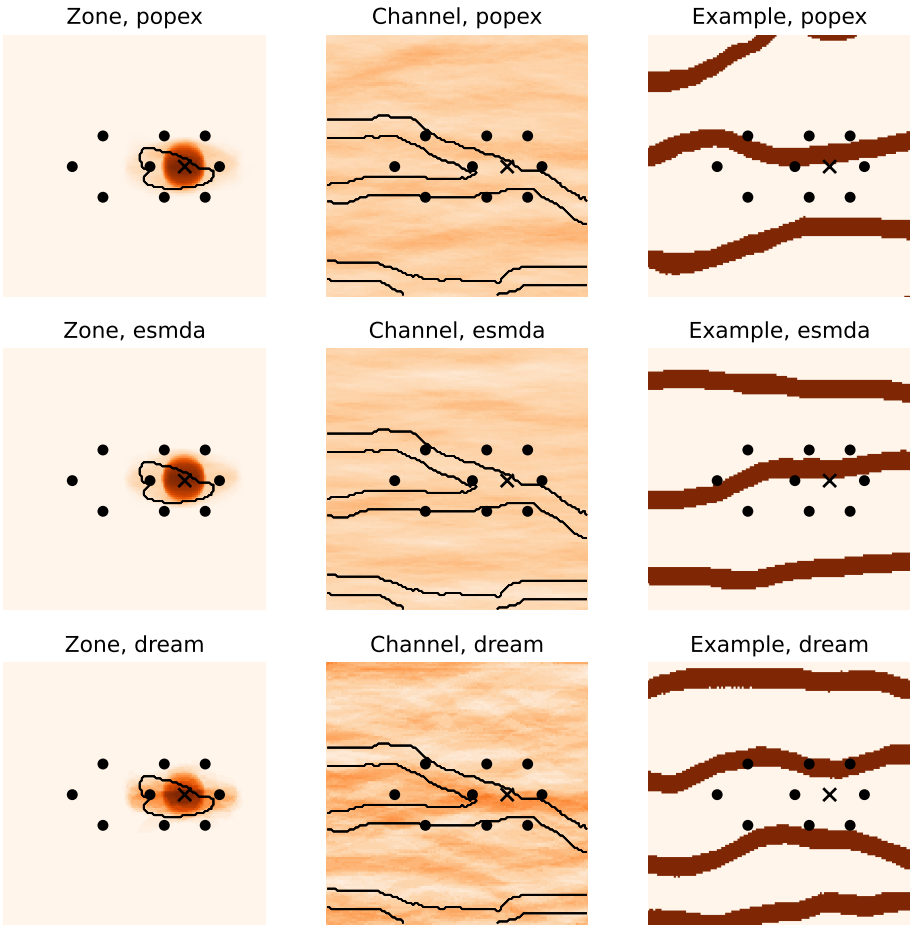

Figure A.20: Prior prediction with Channels TI.

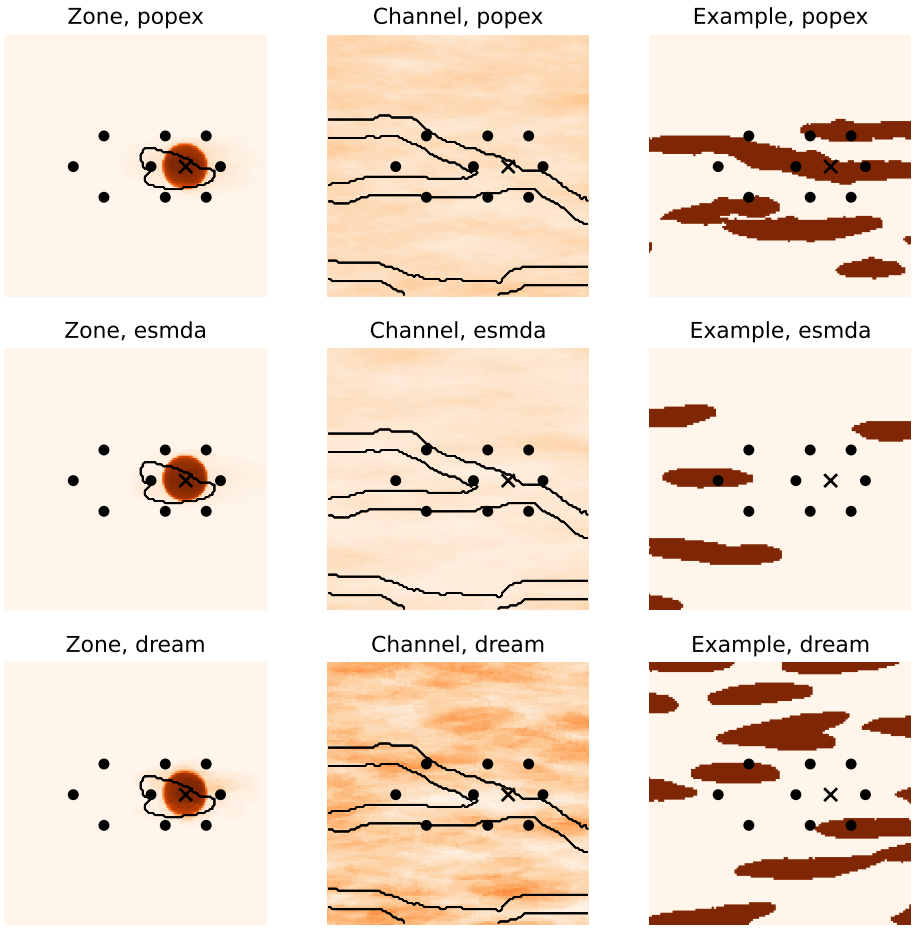

Figure A.21: Prior prediction with Ellipses TI.

Appendix B

Using Generative Adversarial Networks as a Fast Forward Operator for Hydrogeological Inverse Problems[1]

Introduction

Reconstruction of subsurface heterogeneities is a crucial step to make reliable predictions from groundwater flow modelling. Due to the lack of direct observations, inversion techniques are often required to identify the unknown parameters of the system using the observed state data. In hydrogeology, many approaches and solutions to the inverse problem have been proposed. Several review papers (Carrera 1988; Linde et al. 2015; Marsily et al. 2000; Zhou et al. 2014) cover these methods in detail. An important aspect is that it is well known, since the work of Hadamard (1902), that inverse problems are generally ill-posed when they are framed in a deterministic manner: the solution may be non-unique, it may not exist, or it may be unstable. When the inverse problem is framed in a probabilistic manner (Mosegaard and Tarantola 1995; Tarantola and Valette 1982), these issues are not relevant any more. The aim is not to find a unique solution, but instead to obtain the probability distribution of the parameter values that is compatible with the observed state variables, the known physics, and the prior information about the parameters. In addition, the computational burden required to solve the inverse problem can be considered as a stumbling block. Most inverse modelling techniques are iterative and require a large number of forward model runs. Improvements to solve inverse problems often seek to address the above-mentioned issues.

[1]This appendix was published as: Dagasan, Y., Juda, P., & Renard, P. (2020). Using Generative Adversarial Networks as a Fast Forward Operator for Hydrogeological Inverse Problems. Groundwater, 58(6), 938–950. https://doi.org/10.1111/gwat.13005

Deep learning (DL) has gained momentum in recent years and could help to solve the inverse problem efficiently. Applications have shown the power of this technique in various fields (Collobert and Weston 2008; He et al. 2016; Hinton et al. 2012; Xu et al. 2014). It has lately been adopted in the geoscience community as well. As described by Marçais and de Dreuzy (2017), the main motivations to use DL techniques comprise mitigating computational costs, model reduction, category classification and uncertainty quantification. As such, several studies explored adopting DL in geosciences for low dimensional parametrisation (Laloy et al. 2017), inversion (Chan and Elsheikh 2020; Mosser et al. 2019), porous media reconstruction (Mosser et al. 2017, 2018a,b) and generating realistic geological models honouring physical measurements (Dupont et al. 2018). Particular implementations for hydrogeological problems include emulating CPU-demanding reactive transport models (Laloy and Jacques 2019), bidirectional mappings between the parameter space-model state space (Sun 2018) and surrogate modelling (Mo et al. 2019a,b; Tripathy and Bilionis 2018; Zhu and Zabaras 2018).

The purpose of this paper is to investigate the feasibility of using a conditional generative adversarial network (cGAN) as a surrogate model for the forward operator in a hydrogeological probabilistic inversion problem. In distinction from previous surrogate modelling studies using DL (Mo et al. 2019a,b; Tripathy and Bilionis 2018; Yang et al. 2018; Zhu and Zabaras 2018; Zhu et al. 2019), we assess the practical integration of a DL algorithm in the posterior population expansion (PoPEx) algorithm (Jäggli et al. 2018) to perform the inversion in a probabilistic manner. PoPEx is based on the principle of an adaptive importance sampling strategy (Bugallo et al. 2017) but designed specifically for the categorical inverse problem. PoPEx aims at identifying an ensemble of categorical fields representing, for example, the rock types - such as channels and lenses - within an aquifer such that the resulting models would reproduce the observations of state variables within an acceptable error range. In previous papers (Jäggli et al. 2017; Jäggli et al. 2018), it was shown that PoPEx was faster than other Markov Chain Monte Carlo methods to solve the inverse problem in the categorical case. However, as in most inversion algorithms, PoPEx needs a forward operator to compute the state values based on a given parameter field. This is the most time consuming part in the inversion procedure. In this paper, we evaluate the possibility to emulate and replace the forward operator with a conditional generative adversarial network (cGAN). We propose a structure for the cGAN and evaluate the quality of the results when used in the inversion procedure. The tests are made using PoPEx, but the results are more general since the cGAN method could be used in many other inverse approaches.

The rest of the paper is organized as follows. Section 2 provides an overview of the inverse problem formulation, adaptive importance sampling algorithm and the cGAN methods. Section 3 presents the problem setup used in the study and along with the cGAN architecture used. Details about the cGAN training, predictions and the use of the surrogate model in the inversion are given in

Section 4. The paper is concluded in Section 5 with a discussion.

Methods

Inverse Problem Formulation

This section introduces the probabilistic inverse problem formulation following Tarantola (2005). The general aim is to identify the aquifer parameters from a set of state variable observations $\mathbf{d}^{obs}$ (e.g. hydraulic heads, contamination concentration, or discharge rates) and associated measurement errors. In a probabilistic framework, solving the inverse problem means identifying the posterior probability distribution of the model parameters. Using the terminology of Tarantola (2005), a *model* $\mathbf{m} = \{m_1, m_2, \ldots, m_n\}$ is a finite set of parameters which fully describes the physical system of interest. It can be, for example, a map of hydraulic conductivities.

In this paper, we consider the case of a categorical inverse problem: a model $\mathbf{m}$ represents a map in which each cell m_i can only take s possible different values corresponding for example to different rock types. This problem is difficult because most of the inverse methods, relying on partial derivatives or covariances, cannot handle discrete parameters properly (Linde et al. 2015). These models (or maps) can be generated using any geostatistical technique able to simulate categorical values. Each model is then a realization of a complex multidimensional and categorical prior probability distribution denoted $\rho(\mathbf{m})$.

Given a model $\mathbf{m}$, the flow response is computed using a forward operator $\mathbf{g}$ that solves the partial differential equations describing groundwater flow. This mapping allows to forecast the model responses $\mathbf{d} = \mathbf{g}(\mathbf{m})$ at the observation locations. The calculated values $\mathbf{d}$ are used to assess how well a given model $\mathbf{m}$ is able to reproduce the observed data $\mathbf{d}^{obs}$. This is quantified through the likelihood function $L(\mathbf{m})$ that is constructed by assuming a given distribution of errors.

The solution of the inverse problem is then expressed by characterizing the posterior probability $\sigma(\mathbf{m})$ as follows:

$$\sigma(\mathbf{m}) = c\rho(\mathbf{m})L(\mathbf{m}). \tag{B.1}$$

In the above equation, c denotes a normalization constant. Neither $\rho(\mathbf{m})$ nor $L(\mathbf{m})$ have a fully analytic expression and therefore most methods rely on Monte Carlo techniques (Mosegaard and Tarantola 1995).

Posterior Population Expansion (PoPEx)

A detailed presentation of PoPEx is given in Jäggli et al. (2018). Here, we summarize the approach. The method is a modified adaptive importance sampler

(Bugallo et al. 2017). The models $\mathbf{m}_i$ are generated from a proposal probability distribution and weighted according to their importance (see the section **Posterior Prediction**). The proposal distribution is adjusted during the iterations and tends toward the target posterior distribution $\sigma(\mathbf{m})$.

In practice, the algorithm iteratively expands a set of models:

$$\mathcal{M}^k = \{\mathbf{m}_1, \ldots, \mathbf{m}_k\}. \tag{B.2}$$

For each model, the mean-square error between the predicted $g_i(\mathbf{m}_j)$ and the reference values d_i^{obs} is computed

$$\mathrm{MSE}(\mathbf{m}_j) = \frac{1}{n_{obs}} \sum_{i}^{n_{obs}} \left[g_i(\mathbf{m}_j) - d_i^{obs} \right]^2, \tag{B.3}$$

n_{obs} represents the number of observation points and σ the standard deviation of the observation errors. The likelihood is expressed as a function of the mean square error:

$$L(\mathbf{m}_j) = C \exp\left[-\frac{\mathrm{MSE}(\mathbf{m}_j)}{2\sigma^2} \right], \tag{B.4}$$

with the constant C being unknown. This is why, at each iteration, PoPEx uses (and update) a normalized likelihood:

$$\tilde{L}(\mathbf{m}_j) = \frac{L(\mathbf{m}_j)}{\sum_{r=1}^{k} L(\mathbf{m}_r)}, \quad j = 1, \ldots, k. \tag{B.5}$$

To generate a new model $\mathbf{m}_{k+1}$, PoPEx performs a geostatistical simulation with n_k conditioning data points. Three steps are involved.

First, the number of conditioning points n_k is drawn from a uniform distribution over $\{0, 1, \ldots, n_{\max}\}$, where $n_{\max}$ is the maximum number of conditioning points specified by the user.

In the second step, PoPEx identifies the locations and parameter values (rock types or facies) that correspond to high likelihood values. This is done by comparing two probability distribution maps. The first is denoted $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_s\}$, with s the number of facies and $\mathbf{q}_i$ the probability map for the facies i. It describes the prior information based only on the geological data and geostatistical model. The second, $P^k = \{\mathbf{p}_1^k, \ldots, \mathbf{p}_s^k\}$, is updated at each iteration k. It corresponds to the facies probability maps weighted by the normalized likelihood $\tilde{L}(\mathbf{m}_j)$. To identify in a probabilistic manner the link between the flow data and the facies, PoPEx computes the Kullback-Leibler divergence (KLD):

$$D(P^k \| Q) = \sum_{i=1}^{s} p_i^k \log\left(\frac{p_i^k}{q_i} \right). \tag{B.6}$$

In every location, the KLD map shows how different the two distributions are. PoPEx will then place randomly the n_k conditioning data but with a higher probability in regions of high values of the KLD. At those locations, the facies values are then drawn randomly from the local P^k pdf.

The third and last step is the conditional geostatistical simulation. In this paper, PoPEx uses DeeSse (Straubhaar 2019), an efficient implementation of the Direct Sampling multiple-points statistic (MPS) method (Mariethoz et al. 2010a; Mariethoz and Caers 2015).

Posterior Prediction

The parameters of a groundwater flow model are generally identified because the model is needed to make a prediction $f(\mathbf{m})$ of a quantity of interest (e.g. contaminant travel time, drawdown in a place of interest, maximum sustainable pumping rate, etc.). In a probabilistic framework, the aim is to estimate the expected value $\mu = \mathbb{E}_\sigma [f(\mathbf{m})]$ of that prediction. PoPEx, being an importance sampling technique, estimates μ using a weighted average:

$$\mu \approx \sum_{k=1}^{N} \tilde{w}_k f(\mathbf{m}_k). \tag{B.7}$$

N is the number of models in the ensemble, $\tilde{w}_k$ is the weight for model k. By assuming that the conditioning points are sufficiently distant to be considered independent (n_{max} must be small enough), Jäggli et al. (2018) show that it is possible to express the weights from the Q and P^k maps as follows:

$$w_k = L(\mathbf{m}_k) \prod_{j=1}^{n_k} \frac{q_{r(j)}(x_j)}{p_{r(j)}(x_j)}, \tag{B.8}$$

where $r(j)$ returns the category of the pixel at location $\mathbf{x_j}$. Since the weights can be extremely small (weight degeneracy problem), PoPEx raises them to an exponent α chosen such that the number of effective samples $n_e = \frac{(\sum_i^N w_i^\alpha)^2}{\sum_i^N (w_i^\alpha)^2}$ is at least higher than a value specified by the user. The weights are then normalized $\tilde{w}_k = \frac{w_k^\alpha}{\sum_j^N w_j^\alpha}$. This is an approximation that converges to the exact result if the number of samples is sufficiently large.

Conditional Generative Adversarial Networks

GANs were put forward by Goodfellow et al. (2014) and are able to learn an implicit description of the underlying probability distributions of the images in a training data set. The architecture of the GAN comprises two convolutional neural networks: a discriminator D and a generator G. In the original application, the generator is responsible for synthesizing realistic images $G(\mathbf{z})$

based on a latent variable z. The discriminator is responsible for distinguishing the generated images $G(\mathbf{z})$ from the images of the training set $x \sim p_{data}$. To be more specific, the discriminator computes the probability of a synthetic image $G(\mathbf{z})$ coming from the training data distribution $x \sim p_{data}$. While the generator strives for synthesizing more realistic images, the discriminator aims at better identifying the generated images as fake.

In this study, we adopted the pix2pix version of the Conditional Generative Adversarial Networks (cGAN) (Isola et al. 2017), it can map an input image x into an output image y, $G : x \rightarrow y$. The objective function used for the cGAN that was utilized in this study is as follows:

$$\begin{aligned} \mathcal{L}_{cGAN}(G, D) = \mathbb{E}_x \left\| 1 - D(x, G(x)) \right\|_2 \\ + \mathbb{E}_{x,y} \left\| D(x, y) \right\|_2, \end{aligned} \tag{B.9}$$

where the aim is to minimize G and maximize D. x represents the input images and y denotes the corresponding output images. The discriminator is trained to label the real images as 1 and the synthesized images as 0. Hence, if the discriminator performs well, it will output 0 for the synthesized images and 1 for the real ones. Therefore, the overall loss function will have a high value (maximization objective of the discriminator). Similarly, if the generator is good, the discriminator will label the synthesized images $G(x)$ close to 1, which minimizes the loss function. The ultimate goal is to train a generator which produces very realistic images and the discriminator becomes unable to distinguish it from the real ones, producing values close to 0.5. In addition to the loss function defined above, the cGAN we used also includes an L1 type loss comparing the similarity of the generated image with the ground truth. This term ensures the low-frequency correctness of the generated images.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y} |G(x) - y| \tag{B.10}$$

The final objective function then becomes:

$$G^* = \arg\min_G \arg\max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G), \tag{B.11}$$

where λ assigns a weight for the L1 term (low frequency loss).

Numerical Experiment

The general aim of the numerical experiment presented in this paper is to investigate the feasibility of using the conditional GAN as a substitute for the flow simulation to accelerate the solution of the inverse problem using PoPEx. In this context, it is important to consider a synthetic reality case in which the underground is fully known. This allows testing thoroughly the quality of the results obtained with the proposed methodology. In a practical application,

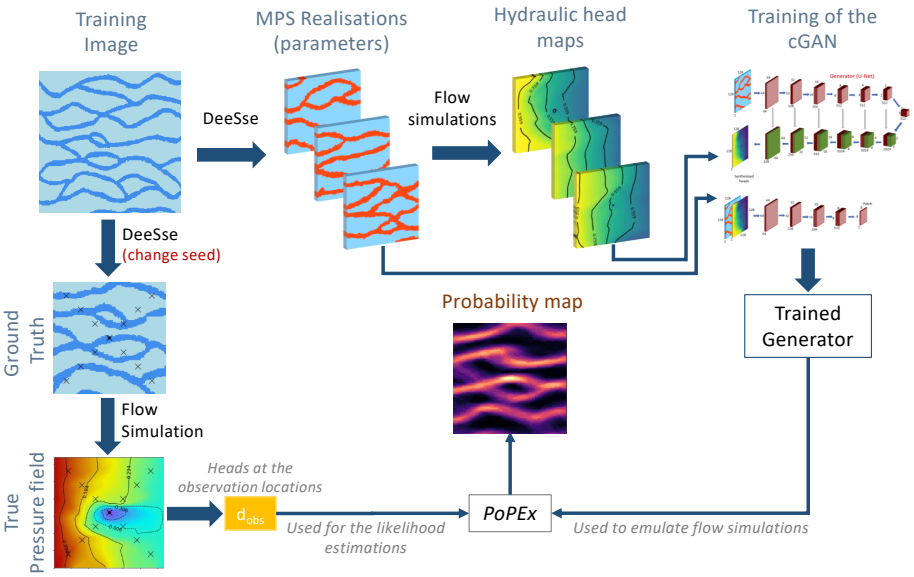

Figure B.1: General overview of the methodology. A channelized training image was used to create a reference true-unknown field and its corresponding flow response. Same training image was then used to create realisations using a different seed value. The created realisations and their corresponding responses were used to train the cGAN model. The trained model was then incorporated in PoPEx to perform the inversion.

the map of the hydraulic conductivity field will never be known exhaustively and it would not be possible to test if the methodology properly identifies the position of the channel and the corresponding uncertainty.

The methodology is illustrated in Fig. B.1. It includes setting up an inverse problem which consists of identifying the position and uncertainty of highly conductive channels from a set of groundwater head measurements around a pumping well. Three inverse methods are applied. One is the most accurate but less efficient (importance sampler based on prior distribution), it provides the reference solution. Then, we apply the standard PoPEx algorithm that uses the flow simulation to evaluate the likelihood. Finally, we apply PoPEx using cGAN to replace the flow simulation. We then compare the accuracy of the results and numerical efficiency.

The detailed explanations for setting up this experiment are presented in the following subsections.

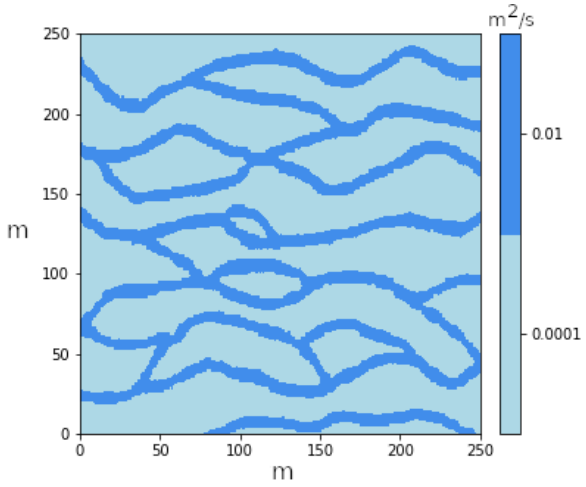

Figure B.2: The training image (TI) used to simulate the aquifer parameters

Parameter	value
Search radius	40
Number of neighbouring nodes	30
Distance threshold	0.02
Maximal scan fraction	0.25
Number of post-processing paths	1

Table B.1: DeeSse parameters used for the model generation.

Problem Setting

The experiment was conducted on a 2D heterogeneous aquifer comprising permeable channels in a less permeable matrix. The spatial structures of the aquifer were simulated using the multiple-point statistical (MPS) simulation tool DeeSse (Straubhaar 2019). The training image (TI) used for the simulations is shown in Fig. B.2 (Strebelle 2002). The DeeSse parameters are provided in Tab. B.1. The size of the TI is 250 $\times$ 250 grid cells and each cell has a dimension of 1 meter in each direction. The two different colours represent two different facies with different uniform hydraulic conductivity (K) values. As for the channels, the hydraulic conductivity value was set to 10^{-2} m/s whereas the matrix was two orders of magnitude less permeable 10^{-4} m/s. The simulation grid comprises 128 grid cells in the x and y directions, and one grid cell in the vertical direction. The thickness of the aquifer is constant and equal to 1 m. Constant head values were applied on the left and right boundaries as 1 m and 0 m, respectively, whereas the upper and lower boundaries had no flow boundary conditions. A pumping well was placed in the centre to extract 3 l/s.

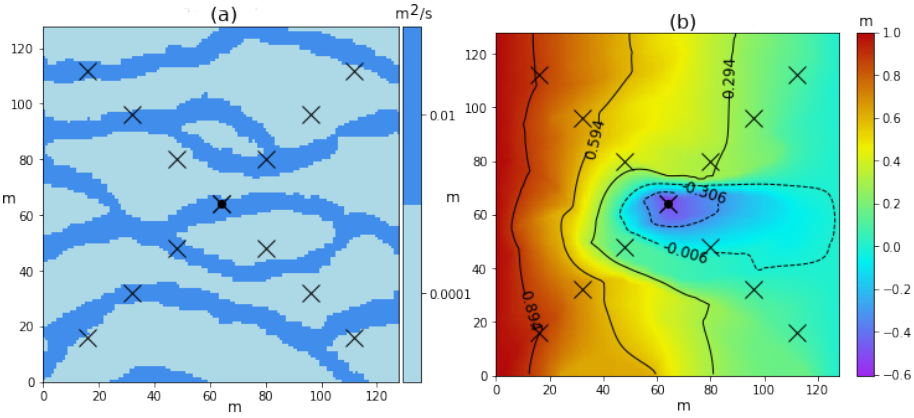

Figure B.3: (a) Reference parameter field that was considered as ground truth and (b) its corresponding flow response.

A reference parameter field was generated using an arbitrary seed and was considered to be the true field. Thirteen measurement locations were determined to extract the hydraulic head values for performing likelihood estimations for the inverse problem. A random Gaussian noise with $\sigma = 0.05$ m was added to the extracted hydraulic head values. The reference parameter field along with its corresponding water heads can be seen in Fig. B.3.

The cGAN was then used to emulate the forward operator that was used to compute the likelihood of a given realization for the parameter field. For training the adversarial network, 600 MPS realizations were performed and the flow simulations for each of the geologies were created using flopy interface of the Modflow software (Bakker et al. 2016). Some examples of the parameter fields and corresponding flow responses can be seen in Fig. B.4.

cGAN Architecture and implementation

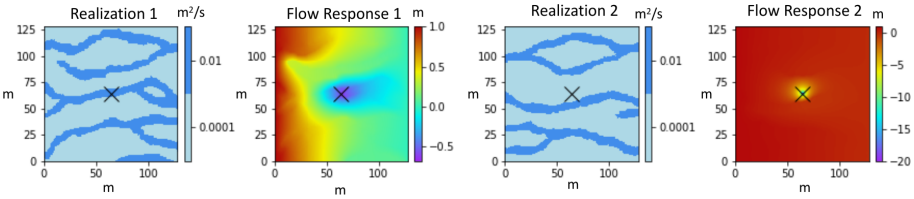

Figure B.4: Some examples from the dataset used to train the cGAN model. Realization 1 and its flow response demonstrate the case in which the well intersects highly permeable channels. Realization 2 and its flow response represent the cases in which the well intersects the low permeability matrix.

The generator utilized the U-Net architecture with skip connections (Reed et al.

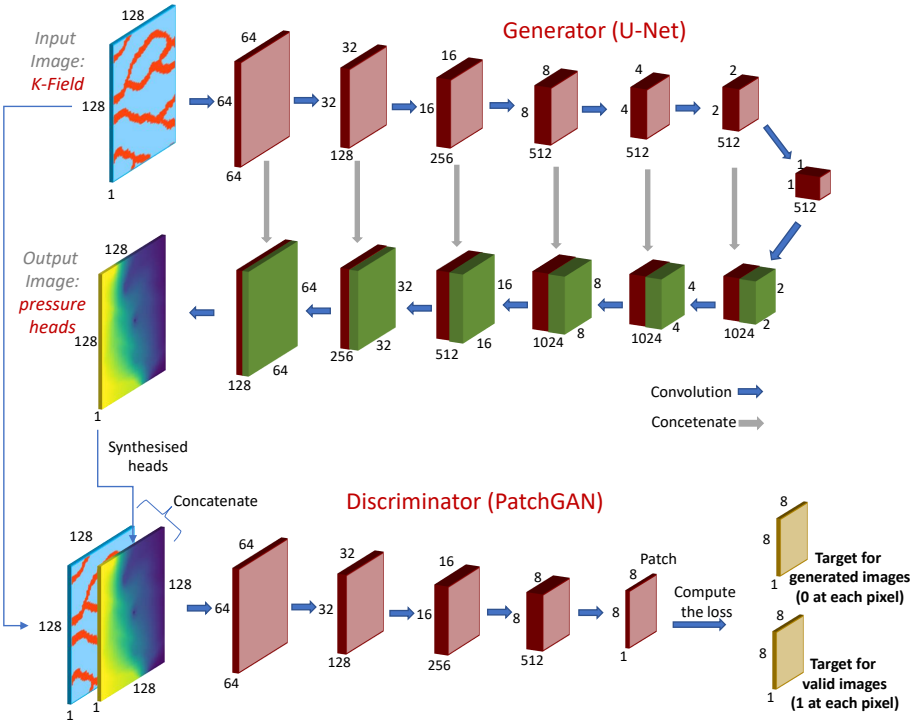

Figure B.5: General overview of the architecture for the generator and discriminator. The discriminator here evaluates the validity of the generated image using the output patch generated.

2016). It comprised an encoder (downsampling) and a decoder (upsampling) parts. Given n total number of layers, the skip connections concatenates the i^{th} layers with those of $n-1$. This structure allows the passing of low-level information between the input and output across the net.

The convolutions performed in both the generator and the discriminator used a 4×4 spatial filter with a stride of 2. The spatial filters are matrices comprising weights and are used to detect/extract features through multiplication operations. The stride corresponds to the step that is used when sliding the filter. The resulting convolved images become the feature maps. Following this step, the feature maps are normalized and the activation function is applied to introduce non-linearity. In this study, the batch normalization was performed with a momentum value of 0.8. As for the activation function, the Rectifier Linear Unit (ReLU) operation (Nair and Hinton 2010) for the encoder was carried out using Leaky ReLU (Maas et al. 2013) with a slope value of 0.2. The decoder did not have a slope value for the leaky ReLU operations. The output layer of the decoder was applied a hyperbolic tangent activation function.

The discriminator used was a Markovian discriminator which computed the texture/style loss. The convolutions were performed using a 4×4 spatial filter with a stride of 2. The convolution process returned an $N \times N$ patch to compute the goodness of the generated images. The target for the real images was an 8×8 patch with values of one at each pixel. Whereas, the target for the generated images was an 8×8 patch with values of zero at each pixel. The parameters of the discriminator were optimized to minimize the difference between the target and the predicted patches. The architecture of the G and D can be seen in Fig. B.5.

The implementation of the methodology was performed using the Keras deep learning library in Python. The sizes of the input and output images (hydraulic conductivity fields and pressure heads, respectively) were 128×128. 500 image pairs were used for training and 100 images were used for testing. Prior to the training, min-max normalization was applied for the training dataset the and values of the images were scaled between 0 and 1. The adam optimization solver was used with a learning rate of 0.0002 and a forgetting rate of 0.5. The number of epochs was 200 and 50 images were used in the minibatches. The λ coefficient for the low frequency loss was set as 100. The training was performed using two Nvidia Tesla K40c GPUs.

Results

cGAN application

To visually check the performance of the trained network, two randomly chosen parameter fields from the test dataset were considered, as can be seen in Fig. B.6. The results in Fig. B.6 give a quick insight on the performance of the cGAN implementation. The results indicate that the generated images are rather similar to their originals and appear to be slightly noisy, as compared to their originals.

Since the trained model is aiming to replace the forward operator within an inversion process in this study, quantitative assessment of the performances were also carried out. Within PoPEx, the likelihood estimation is performed based on the error between the observed and the simulated heads. Therefore, MSE and the coefficient of determination (R^2) values of the generated pressure fields against the actual head maps were used to assess the cGAN performance. 100 test images that were not included in the training set were used to perform the predictions using the trained cGAN model. The predicted head maps were then used to compute the associated coefficient of determination scores (R^2). The mean of the R^2 score was 0.97. This also indicates that the trained cGAN model appears to perform well.

As with many deep learning algorithms, it takes a fair amount of time to train a model. For 200 epochs, a minibatch size of 50, and 500 training images in

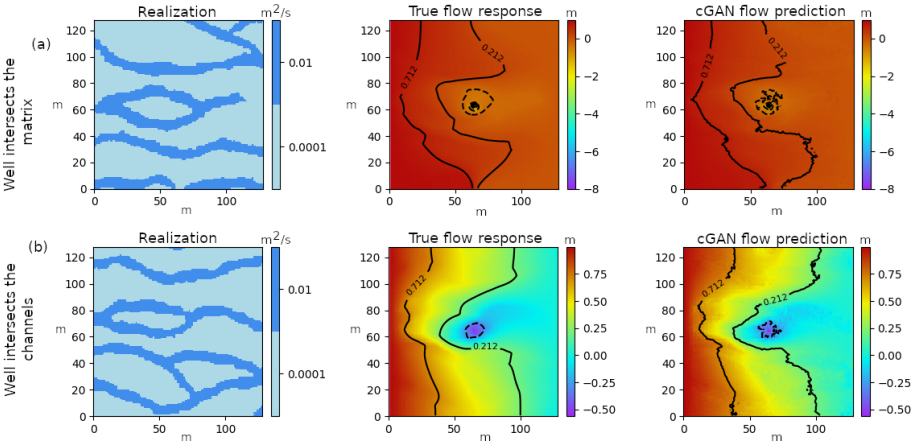

Figure B.6: Validation images to visually check the prediction performance of the trained adversarial network: (a) flow simulation and the cGAN prediction for the given realization (well location intersects the matrix), (b) flow simulation and the cGAN prediction for the given realization (well location intersects the channel).

total, the training times were: about 5.5 hours for the 128x128 cells problem, and 17.3 hours for the 256x256 cells problem. We have further investigated if there would be a potential avenue to reduce the computation time at a cost of reasonable accuracy loss. Our investigation consisted of changing the number of epochs and training images used and observing the errors in terms of MSE.

The MSE obtained for different number of epochs and training images used during training are shown in Fig. B.7. The MSE observed with 200 epochs and 500 training dataset size was found as 0.009. Whereas, mean and standard deviations of the proxy errors were 0.018 and 0.028, respectively. These values indicate that the errors are small when compared to the assumed data error used in the likelihood.

The scatterplots of the actual flow simulations against the predicted ones for different epochs and the training sizes are shown in Fig. B.8 and Fig B.9. Since the response of the flow simulations are different depending on whether the well intersects a channel or not, we provide both cases. The results show that as the number of epochs increases from 100 to 300, the error decreases. However, the MSE starts to significantly increase from 0.009 to 0.10 after epoch number 300. This behaviour could be due to two reasons. First, the discriminator could become relatively more powerful than the generator with an increasing number of epochs. Hence, the generator faces difficulty in fooling the discriminator. One solution would be to modify the learning rates of the discriminator. Secondly, the discriminator might have undergone an overfitting issue. A pos-

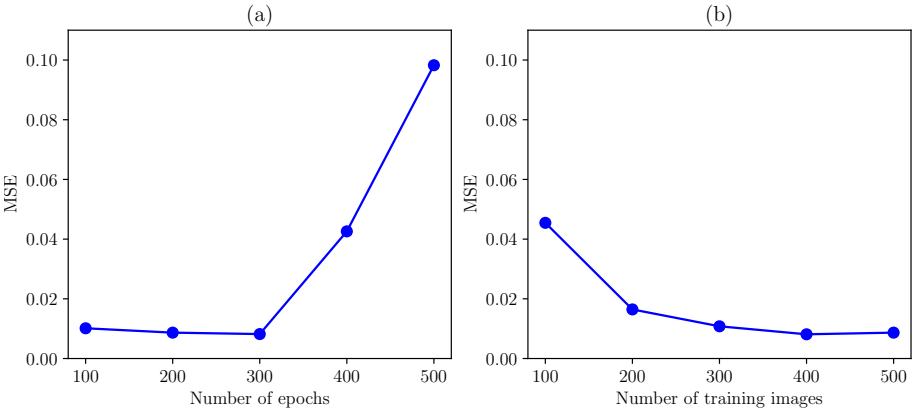

Figure B.7: Mean squared error between the actual and the cGAN predicted head maps based on (a) different number of epochs and (b) different training dataset size.

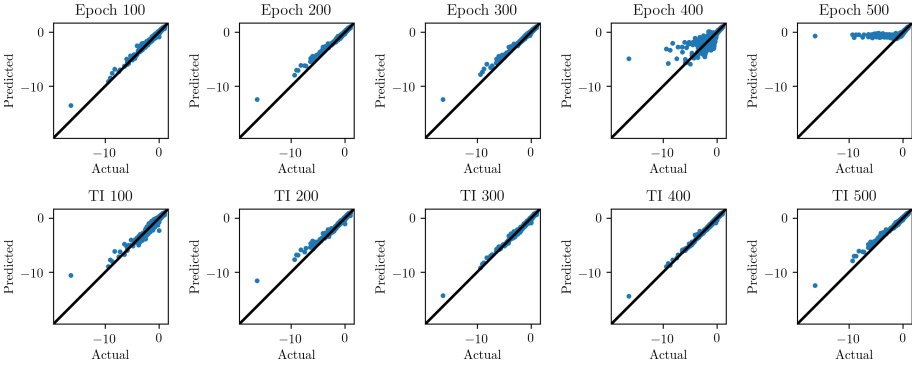

Figure B.8: Scatterplot of the actual head values against the cGAN predictions for different epochs and training set sizes. Plots belong to the flow simulations in which the pumping well intersects the matrix.

sible solution to avoid this could be adding Gaussian noise to the input images. Nevertheless, since we have already obtained sufficient accuracies with epoch numbers less than 300, we have not further investigated this.

In the light of the above information, to lower the CPU cost, the model can also be trained using a reduced number of epochs such as 100 at a cost of reasonable accuracy loss. As for the number of training images, the error decreases with the increasing number of training images, as expected. Similarly, the training time can be reduced using a reduced training size of 300-400 by sacrificing reasonable accuracy.

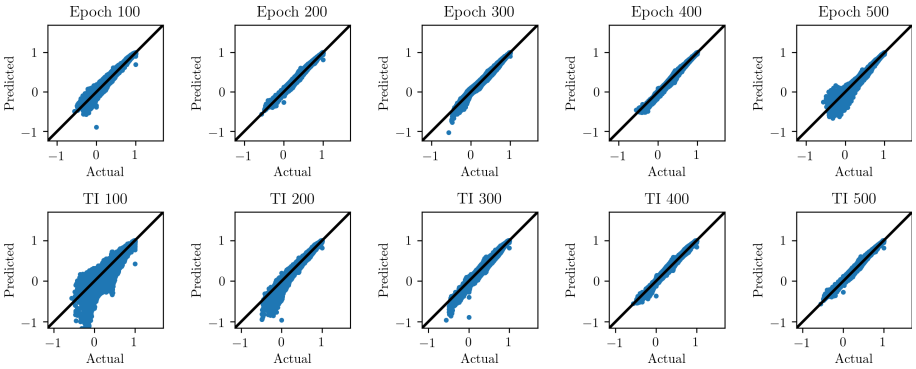

Figure B.9: Scatterplot of the actual head values against the cGAN predictions for different epochs and training set sizes. Plots belong to the flow simulations in which the pumping well intersects the channels.

A test was performed to compare the processing power demand of the cGAN model with the Modflow engine. In order to have a fair comparison between the two, we have performed a single CPU estimation of 500 head fields and repeated it ten times. In addition, we have created a similar setup with 256 $\times$ 256 grid resolutions. Similarly, we have trained the cGAN with 500 image pairs and performed the predictions. This allowed us to see the effect of the number of parameters in the conductivity field on both the cGAN and Modflow simulations. The CPU used was i7-8750H and the GPU used for the predictions was GeForce GTX 1050. The results in Fig. B.10 demonstrate that cGAN required less computational power as compared to Modflow in both resolutions. Considering the tests done on single CPUs, we have observed 51% and 30% reduction in the computation times for the 128 $\times$ 128 and 256 $\times$ 256 resolutions, respectively. The same test done on the GPU, revealed a significant improvement. Use of GPU demonstrated an 83% computational reduction for the 128 $\times$ 128 resolution and 79% reduction for the 256 $\times$ 256 resolutions.

There is 105 seconds time difference between Modflow and GPU processing times to compute 500 flow simulations.

PoPEx Results

To test the performance of the GAN-flow emulator, we used it as the forward operator in the PoPEx procedure. We then compared the PoPEx solution using the cGAN forward emulator with two alternatives: the reference sampler and the PoPEx solution with Modflow forward simulator. Fig. B.11 shows the posterior probability maps of the channel for the three set-ups. The prior probability map for the channel is uniform over the entire domain with the value of 0.72.

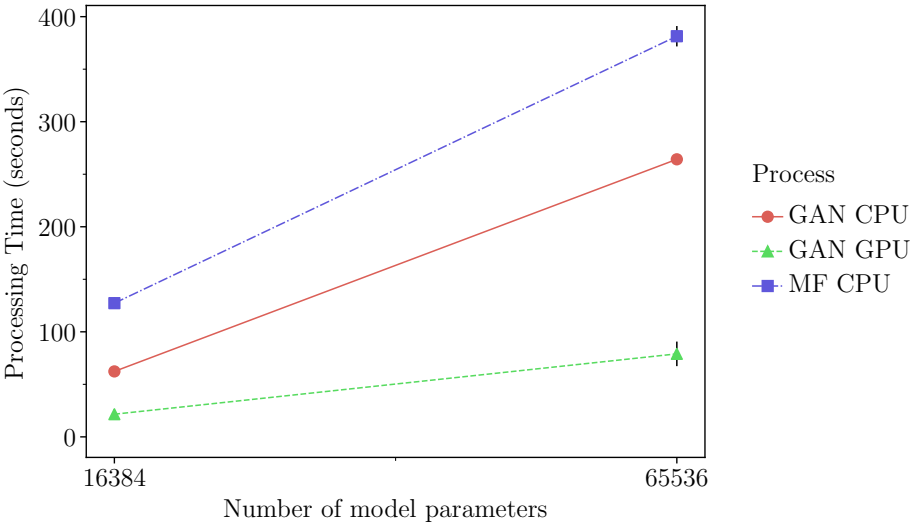

Figure B.10: Average processing time required for running the forward operator on 500 flow problems as a function of the number of parameters in the input parameter field.

To compute the reference solution, we applied the importance sampling method using the prior distribution as the proposal distribution (Cary and Chapman 1988). In that case, the probability map is obtained using Eqn. (B.7) with the weights being simply the normalized likelihood values (Jäggli et al. 2018). This method is not the most efficient but since we applied it for a large number of models (300,000) and because the problem is of relatively small size, we ensure high accuracy of the result.

Model generation in all the cases was done by the DeeSse algorithm with the same parameter set defined in Tab. B.1, which was also used for training of the GAN-flow emulator. The PoPEx solutions (both with GAN-flow and with Modflow) are based on 10,000 iterations, the minimal number of models used for prediction is $l_0 = 100$ and maximal number of conditioning points guiding simulations is 20.

The results given in Fig. B.11 show that both PoPEx solutions correctly reproduced the channelized structure of the probability map. The quality of the PoPEx with GAN-flow solution is visually comparable to that of PoPEx with Modflow, only slight underestimation of the uncertainty of having a channel (stronger contrasts), is visible on the right side of the image.

To quantify the error of the PoPEx solution with respect to the prior sampling solution, we used the Jensen-Shannon divergence. Considering the posterior solution $\hat{\mu}$, the Jensen-Shannon divergence with respect to the reference prior

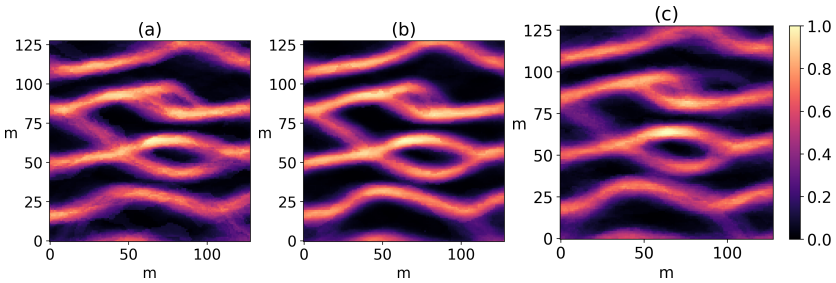

Figure B.11: Comparison of posterior facies probability maps for channel, obtained using: (a) the reference technique (300,000 iterations), (b) PoPEx with Modflow forward (10,000 iterations) and (c) PoPEx with GAN-flow forward (10,000 iterations).

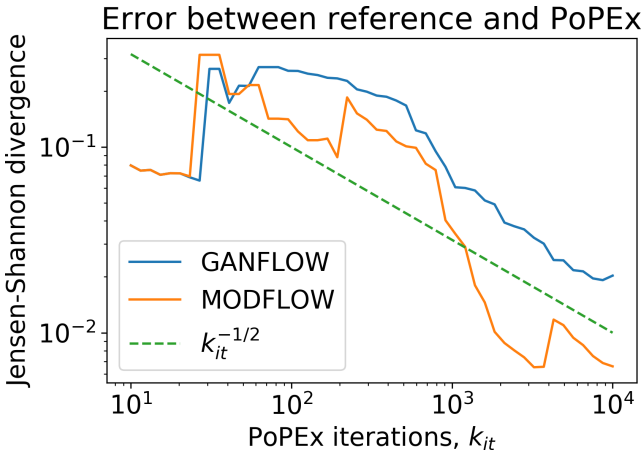

Figure B.12: Comparison of the error of the inverse problem solution with respect to the prior sampling solution as a function of the number of PoPEx iterations k_{it} for the two different forward simulators. Error is defined by average Jensen-Shannon divergence.

sampling solution μ^{ex} is given by:

$$J(\hat{\mu}||\mu^{ex}) = \frac{1}{2}\left[D\left(\hat{\mu}||\bar{\mu}\right) + D\left(\mu^{ex}||\bar{\mu}\right)\right], \tag{B.12}$$

where $D(\cdot||\cdot)$ is the Kullback-Leibler divergence as defined in Equation (B.6) and $\bar{\mu} = \frac{1}{2}(\hat{\mu} + \mu^{ex})$. The errors for the PoPEx solutions $\hat{\mu}$ using GAN-flow and Modflow as a function of the number of iterations is plotted in Fig. B.12.

For the solution obtained after 10,000 iterations, we obtained the following values of the averaged Jensen-Shannon divergence: 0.007 for Modflow and 0.021 for GAN-flow. The maps of errors given by Jensen-Shannon divergence for

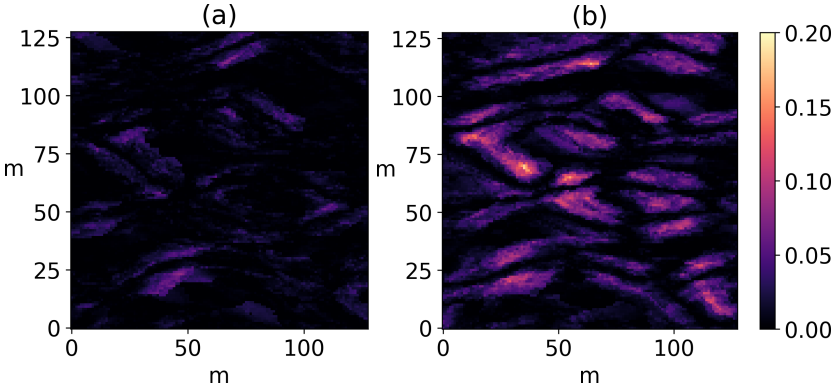

Figure B.13: Maps of pixel-wise errors of PoPEx solution obtained after 10,000 iterations: (a) using Modflow, (b) using GAN-flow.

both flow simulators are plotted in Fig. B.13. The convergence plot (Fig. B.12) demonstrates that GAN-flow error goes down systematically with the Modflow error. Even if the GAN-flow error is slightly bigger than the Modflow error, both convergence rates compare well with the theoretical rate of $k_{it}^{-1/2}$ (where k_{it} is the number of iterations) predicted by the Central Limit Theorem (Durrett 2019).

Discussion and Conclusion

We have implemented conditional Generative Adversarial Networks to emulate the forward operator used in a hydrogeological inverse problem. The model was trained using a dataset comprising 1) 500 hydraulic conductivity fields that were created using MPS simulations and 2) corresponding flow simulations obtained using Modflow. Having trained the cGAN, it was then used to emulate the forward operator of the PoPEx algorithm to compute the likelihoods. Prediction performances were assessed using a separate test dataset containing 100 parameter fields and corresponding flow simulations.

The results show that the cGAN is capable of mapping from a given hydraulic conductivity field to the corresponding flow simulations. Although the generated pressure maps are slightly noisy as compared to their referenced ones, there exists a high correlation between the predicted and actual head values. These errors were sufficiently small to have little impact on the results of the probabilistic inversion. The probability maps obtained using the cGAN were very close to the ones obtained using the exact forward solution. As compared to the PoPEx solution, the cGAN produced slightly less contrast at some locations. Nevertheless, this experiment demonstrated the capability of the cGAN to be used in the inversion successfully with less computational demand than the Modflow engine.

The use of cGAN is of course not limited to PoPEx. Any inverse method that uses repeatedly the forward model to evaluate the misfit could benefit from this technique. The gain will be the highest for inverse methods that make intensive use of the forward model such as Markov chain Monte Carlo methods (Mosegaard and Tarantola 1995), Particle or Ensemble Kalman Filters (Evensen 2018).

However, there are several drawbacks. First, as in most deep learning algorithms, training a model requires the creation of a training dataset. For complex geologies with numerous grid cells, this could take significant time. For such cases, a smaller sized dataset containing a smaller number of training instances can be still be used to perform reasonable predictions at the cost of some accuracy loss. Second, training of GANs can require significant time. To compensate for the training times, in the example that was treated in this paper, we require roughly 100,000 iterations of PoPEx. In other words, the use of cGAN becomes advantageous only after around a hundred thousand iterations. Considering that PoPEx can provide an accurate solution already after 10,000 iterations, we observe that the required number of iterations to compensate the training is quite large. This may hinder its use in practical applications. One solution could be to find some ways to simplify the generator network architecture to have fewer layers and to ensure faster training. This will probably reduce the quality of the approximation (Mallat 2016). However, for the specific application discussed in this paper, we do not necessarily need to predict the complete hydraulic head field. A simpler surrogate model could be sufficient to predict the likelihood with enough accuracy and be much faster. Another solution could be the use of transfer learning (Pan and Yang 2009). A pre-trained model on a similar problem can be used to retrieve the initial parameters of the network architecture. The subsequent training would then involve fine-tuning of the parameters in a shorter period of training time for the specific application (Ng et al. 2015). These techniques have been shown to be very efficient.

To conclude, this paper shows the strength of the cGAN method in general but also highlights some practical limitations that are not frequently discussed in the literature concerning the application of these methods for groundwater modelling. It emphasizes the need to conduct further research in this field.

Acknowledgments

The authors would like to thank Julien Straubhaar and Christoph Jäggli for their help and suggestions concerning this work. We also thank the editor Henk M. Haitjema, two anonymous reviewers and Jean Marçais for their valuable comments and suggestions. We also appreciate Boris Galvan help in providing an access to the GPU cluster. Yasin Dagasan is grateful for the funding by the Swiss Government Excellence Scholarship.

Bibliography

Abdollahifard, Mohammad Javad, Grégoire Mariéthoz, and Maryam Ghavim (2019). "Quantitative Evaluation of Multiple-Point Simulations Using Image Segmentation and Texture Descriptors". In: *Computational Geosciences* 23.6, pp. 1349–1368. ISSN: 1420-0597, 1573-1499.

Alcolea, Andres and Philippe Renard (2010). "Blocking Moving Window Algorithm: Conditioning Multiple-Point Simulations to Hydrogeological Data". In: *Water Resources Research* 46.8. ISSN: 1944-7973.

Allard, Denis, Alessandro Comunian, and Philippe Renard (2012). "Probability Aggregation Methods in Geoscience". In: *Mathematical Geosciences* 44.5, pp. 545–581.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, pp. 214–223.

Arlot, Sylvain and Alain Celisse (2010). "A Survey of Cross-Validation Procedures for Model Selection". In: *Statist. Surv.* 4, pp. 40–79.

Arnold, Dan, Vasily Demyanov, Temistocles Rojas, and Mike Christie (2019). "Uncertainty Quantification in Reservoir Prediction: Part 1—Model Realism in History Matching Using Geological Prior Definitions". In: *Mathematical Geosciences* 51.2, pp. 209–240.

Bakker, M., V. Post, C. D. Langevin, J. D. Hughes, J. T. White, J. J. Starn, and M. N. Fienen (2016). "Scripting MODFLOW Model Development Using Python and FloPy". In: *Groundwater* 54.5, pp. 733–739.

Baninajar, Ehsanollah, Yousef Sharghi, and Gregoire Mariethoz (2019). "MPS-APO: A Rapid and Automatic Parameter Optimizer for Multiple-Point Geostatistics". In: *Stochastic Environmental Research and Risk Assessment* 33.11-12, pp. 1969–1989. ISSN: 1436-3240, 1436-3259.

Boisvert, Jeff B., Michael J. Pyrcz, and Clayton V. Deutsch (2007). "Multiple-Point Statistics for Training Image Selection". In: *Natural Resources Research* 16.4, pp. 313–321. ISSN: 1573-8981.

Breiman, Leo (2001). "Random Forests". In: *Machine Learning* 45.1, pp. 5–32.

Breiman, Leo, Jerome Friedman, C.J. Stone, and R.A. Olshen (1984). *Classification and Regression Trees*. The Wadsworth and Brooks-Cole Statistics-Probability Series. Taylor & Francis. ISBN: 978-0-412-04841-8.

Breiman, Leo and Philip Spector (1992). "Submodel Selection and Evaluation in Regression. The X-Random Case". In: *International Statistical Review / Revue Internationale de Statistique* 60.3, pp. 291–319. ISSN: 03067734, 17515823.

Brier, Glenn W. (1950). "Verification Of Forecasts Expressed In Terms Of Probability". In: *Monthly Weather Review* 78.1, pp. 1–3.

Brodersen, Kay Henning, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann (2010). "The Balanced Accuracy and Its Posterior Distribution". In: *2010 20th International Conference on Pattern Recognition*. IEEE, pp. 3121–3124.

Bugallo, Monica F., Victor Elvira, Luca Martino, David Luengo, Joaquin Miguez, and Petar M. Djuric (2017). "Adaptive Importance Sampling: The past, the present, and the future". en. In: *IEEE Signal Processing Magazine* 34.4, pp. 60–79.

Caers, Jef and Todd Hoffman (2006). "The Probability Perturbation Method: A New Look at Bayesian Inverse Modeling". In: *Mathematical geology* 38.1, pp. 81–100.

Canchumuni, Smith W A (2021). "Recent Developments Combining Ensemble Smoother and Deep Generative Networks for Facies History Matching". In: *Comput Geosci*, p. 34.

Carrera, Jesus (1988). "State of the art of the inverse problem applied to the flow and solute transport equations". In: *Groundwater flow and quality modelling. NATO ASI Series (Series C: Mathematical and Physical Sciences)*. Ed. by E. Custodio, A. Gurgui, and J.P.L. Ferreira. Vol. 224. Dordrecht: Springer, pp. 549–583.

Cary, PW and CH Chapman (1988). "Automatic 1-D waveform inversion of marine seismic refraction data". In: *Geophysical Journal International* 93.3, pp. 527–546.

Chan, Shing and Ahmed H. Elsheikh (2020). "Parametrization of Stochastic Inputs Using Generative Adversarial Networks With Application in Geology". In: *Frontiers in Water* 2, p. 5. ISSN: 2624-9375.

Chiles, Jean-Paul and Pierre Delfiner (2012). *Geostatistics: Modeling Spatial Uncertainty, 2nd Edition*. Wiley.

Collobert, Ronan and Jason Weston (2008). "A unified architecture for natural language processing: Deep neural networks with multitask learning". In: *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 160–167.

Cressie, Noel A. C. (1993). *Statistics for Spatial Data*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc.

Dagasan, Yasin, Przemysław Juda, and Philippe Renard (2020). "Using Generative Adversarial Networks as a Fast Forward Operator for Hydrogeological Inverse Problems". In: *Groundwater* 58.6, pp. 938–950. ISSN: 1745-6584.

Dagasan, Yasin, Philippe Renard, Julien Straubhaar, Oktay Erten, and Erkan Topal (2018). "Automatic Parameter Tuning of Multiple-Point Statistical Simulations for Lateritic Bauxite Deposits". In: *Minerals* 8.5, p. 220.

Dall'Alba, Valentin, Philippe Renard, Julien Straubhaar, Benoit Issautier, Cédric Duvail, and Yvan Caballero (Oct. 26, 2020). "3D Multiple-Point Statistics Simulations of the Roussillon Continental Pliocene Aquifer Using DeeSse". In: *Hydrology and Earth System Sciences* 24.10, pp. 4997–5013. ISSN: 1607-7938.

David, M (1976). "The Practice of Kriging". In: *Advanced Geostatistics in the Mining Industry*. Springer, pp. 31–48.

Delfiner, Pierre (1976). "Linear Estimation of Non Stationary Spatial Phenomena". In: *Advanced Geostatistics in the Mining Industry*. Springer, pp. 49–68.

Demyanov, Vasily, Alexei Pozdnoukhov, Mike Christie, and Mikhail Kanevski (2010). "Detection of Optimal Models in Parameter Space with Support Vector Machines". In: *geoENV VII–Geostatistics for Environmental Applications*. Springer, pp. 345–358.

Dubrule, Olivier (1983). "Cross Validation of Kriging in a Unique Neighborhood". In: *Mathematical Geology* 15.6, pp. 687–699.

Dupont, Emilien, Tuanfeng Zhang, Peter Tilke, Lin Liang, and William Bailey (2018). "Generating realistic geology conditioned on physical measurements with generative adversarial networks". In: *arXiv preprint arXiv:1802.03065*.

Durrett, Rick (2019). *Probability: theory and examples*. Vol. 49. Cambridge university press.

Emerick, Alexandre A. (2016). "Analysis of the Performance of Ensemble-Based Assimilation of Production and Seismic Data". In: *Journal of Petroleum Science and Engineering* 139, pp. 219–239. ISSN: 09204105.

Emerick, Alexandre A. and Albert C. Reynolds (2012). "History Matching Time-Lapse Seismic Data Using the Ensemble Kalman Filter with Multiple Data Assimilations". In: *Computational Geosciences* 16.3, pp. 639–659. ISSN: 1573-1499.

— (June 1, 2013). "Ensemble Smoother with Multiple Data Assimilation". In: *Computers & Geosciences* 55, pp. 3–15. ISSN: 0098-3004.

Evensen, Geir (2018). "Analysis of iterative ensemble smoothers for solving inverse problems". In: *Computational Geosciences* 22.3, pp. 885–908.

Feng, Wenjie, Shenghe Wu, Yanshu Yin, Jiajia Zhang, and Ke Zhang (2017). "A Training Image Evaluation and Selection Method Based on Minimum Data Event Distance for Multiple-Point Geostatistics". In: *Computers & Geosciences* 104, pp. 35–53. ISSN: 0098-3004.

Freund, Yoav and Robert E. Schapire (1997). "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1, pp. 119–139. ISSN: 0022-0000.

Gaspari, Gregory and Stephen E. Cohn (1999). "Construction of Correlation Functions in Two and Three Dimensions". In: *Quarterly Journal of the Royal Meteorological Society* 125.554, pp. 723–757. ISSN: 1477-870X.

Geisser, Seymour (Apr. 1974). "A Predictive Approach to the Random Effect Model". In: *Biometrika* 61.1, pp. 101–107. ISSN: 0006-3444.

— (1975). "The Predictive Sample Reuse Method with Applications". In: *Journal of the American Statistical Association* 70.350, pp. 320–328.

Gelman, Andrew and Donald B. Rubin (Nov. 1992). "Inference from Iterative Simulation Using Multiple Sequences". In: *Statistical Science* 7.4, pp. 457–472. ISSN: 0883-4237, 2168-8745.

Gneiting, Tilmann, Fadoua Balabdaoui, and Adrian E Raftery (2007). "Probabilistic Forecasts, Calibration and Sharpness". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69.2, pp. 243–268.

Gneiting, Tilmann and Adrian E. Raftery (2007). "Strictly Proper Scoring Rules, Prediction, and Estimation". In: *Journal of the American Statistical Association* 102.477, pp. 359–378.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. Cambridge, MA: MIT Press.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adversarial nets". In: *Advances in neural information processing systems*, pp. 2672–2680.

Gravey, Mathieu and Grégoire Mariethoz (June 8, 2020). "QuickSampling v1.0: A Robust and Simplified Pixel-Based Multiple-Point Simulation Approach". In: *Geoscientific Model Development* 13.6, pp. 2611–2630. ISSN: 1991-959X.

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville (2017). "Improved Training of Wasserstein GANs". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc.

Hadamard, Jacques (1902). "Sur les problèmes aux dérivées partielles et leur signification physique". In: *Princeton university bulletin*, pp. 49–52.

Hansen, Thomas Mejer, Knud Skou Cordua, and Klaus Mosegaard (2012). "Inverse Problems with Non-Trivial Priors: Efficient Solution through Sequential Gibbs Sampling". In: *Computational Geosciences* 16.3, pp. 593–611.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hendricks Franssen, Harrie-Jan, Andrés Alcolea, Monica Riva, Mahmoud Bakr, N Van der Wiel, Fritz Stauffer, and Alberto Guadagnini (2009). "A Comparison of Seven Methods for the Inverse Modelling of Groundwater Flow. Application to the Characterisation of Well Catchments". In: *Advances in Water Resources* 32.6, pp. 851–872.

Hinton, Geoffrey, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. (2012). "Deep neural networks for acoustic modeling in speech recognition". In: *IEEE Signal processing magazine* 29.

Hughes, Joseph D., Christian D. Langevin, and Edward R. Banta (2017). *Documentation for the MODFLOW 6 Framework*. Report 6-A57. Reston, VA.

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros (2017). "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134.

Jäggli, C, J Straubhaar, and Philippe Renard (2017). "Posterior Population Expansion for Solving Inverse Problems". In: *Water Resources Research* 53.4, pp. 2902–2916.

Jäggli, Christoph, Julien Straubhaar, and Philippe Renard (2018). "Parallelized Adaptive Importance Sampling for Solving Inverse Problems". In: *Frontiers in Earth Science* 6, p. 203. ISSN: 2296-6463.

Juda, P., P. Renard, and J. Straubhaar (2019). "K-Fold Cross-validation of Multiple-point Statistical Simulations". In: Petroleum Geostatistics 2019. Vol. 2019. 1. European Association of Geoscientists & Engineers, pp. 1–5.

Juda, Przemysław, Philippe Renard, and Julien Straubhaar (2020). "A Framework for the Cross-Validation of Categorical Geostatistical Simulations". In: *Earth and Space Science* 7.8, e2020EA001152. ISSN: 2333-5084.

Kang, Xueyuan, Xiaoqing Shi, André Revil, Zhendan Cao, Liangping Li, Tian Lan, and Jichun Wu (2019). "Coupled Hydrogeophysical Inversion to Identify Non-Gaussian Hydraulic Conductivity Field by Jointly Assimilating Geochemical and Time-Lapse Geophysical Data". In: *Journal of Hydrology* 578, p. 124092.

Kelleher, John D., Brian Mac Namee, and Aoife D'Arcy (2015). *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. The MIT Press.

Kitanidis, Peter K. (1991). "Orthonormal Residuals in Geostatistics: Model Criticism and Parameter Estimation". In: *Mathematical Geology* 23.5, pp. 741–758. ISSN: 1573-8868.

Kohavi, Ron (1995). "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection". In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2* (Montreal, Quebec, Canada). IJCAI'95. Morgan Kaufmann Publishers Inc., pp. 1137–1143. ISBN: 1-55860-363-8.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. Curran Associates, Inc., pp. 1097–1105.

Laloy, Eric, Romain Hérault, Diederik Jacques, and Niklas Linde (2018). "Training-Image Based Geostatistical Inversion Using a Spatial Generative Adversarial Neural Network". In: *Water Resources Research* 54.1, pp. 381–406.

Laloy, Eric, Romain Hérault, John Lee, Diederik Jacques, and Niklas Linde (2017). "Inversion Using a New Low-Dimensional Representation of Complex Binary Geological Media Based on a Deep Neural Network". In: *Advances in Water Resources* 110, pp. 387–405. ISSN: 0309-1708.

Laloy, Eric and Diederik Jacques (2019). "Emulation of CPU-demanding Reactive Transport Models: A Comparison of Gaussian Processes, Polynomial Chaos Expansion, and Deep Neural Networks". In: *Computational Geosciences* 23.5, pp. 1193–1215.

Laloy, Eric, Niklas Linde, Cyprien Ruffino, Romain Hérault, Gilles Gasso, and Diederik Jacques (2019). "Gradient-Based Deterministic Inversion of Geophysical Data with Generative Adversarial Networks: Is It Feasible?" In: *Computers & Geosciences* 133, p. 104333. ISSN: 0098-3004.

Laloy, Eric and Jasper A. Vrugt (2012). "High-Dimensional Posterior Exploration of Hydrologic Models Using Multiple-Try DREAM(ZS) and High-Performance Computing". In: *Water Resources Research* 48.1. ISSN: 1944-7973.

Lam, D. T. (2019). "Conditioning Groundwater Flow Parameters with Iterative Ensemble Smoothers: Analysis and Approaches in the Continuous and the Discrete Cases". Neuchâtel: University of Neuchâtel.

Lam, D.-T., P. Renard, J. Straubhaar, and J. Kerrou (2020). "Multiresolution Approach to Condition Categorical Multiple-Point Realizations to Dynamic Data With Iterative Ensemble Smoothing". In: *Water Resources Research* 56.2, e2019WR025875. ISSN: 1944-7973.

LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel (1989). "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural computation* 1.4, pp. 541–551.

Li, Liangping and Fleford Redoloza (2020). "Coupling Ensemble Smoother and Deep Learning with Generative Adversarial T Networks to Deal with Non-Gaussianity in Flow and Transport Data Assimilation". In: *Journal of Hydrology*, p. 16.

Linde, Niklas, Philippe Renard, Tapan Mukerji, and Jef Caers (2015). "Geological Realism in Hydrogeological and Geophysical Inverse Modeling: A Review". In: *Advances in Water Resources* 86, pp. 86–101. ISSN: 0309-1708.

Maas, Andrew L, Awni Y Hannun, and Andrew Y Ng (2013). "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. icml*. Vol. 30. 1, p. 3.

Madani, Nasser, Mohammad Maleki, and Xavier Emery (2018). "Nonparametric Geostatistical Simulation of Subsurface Facies: Tools for Validating the Reproduction of, and Uncertainty in, Facies Geometry". In: *Natural Resources Research*.

Maharaja, Amisha (2008). "TiGenerator: Object-based training image generator". In: *Computers & Geosciences* 34.12, pp. 1753–1761. ISSN: 0098-3004.

Mallat, Stéphane (2016). "Understanding deep convolutional networks". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065, p. 20150203.

Marçais, Jean and Jean-Raynald de Dreuzy (2017). "Prospective Interest of Deep Learning for Hydrological Inference". In: *Groundwater* 55.5, pp. 688–692.

Mariethoz, G., P. Renard, and J. Straubhaar (2010a). "The Direct Sampling method to perform multiple-point geostatistical simulations". In: *Water Resources Research* 46.W11536.

Mariethoz, Grégoire (2010). "A General Parallelization Strategy for Random Path Based Geostatistical Simulation Methods". In: *Computers & Geosciences* 36.7, pp. 953–958. ISSN: 0098-3004.

Mariethoz, Gregoire and Jef Caers (2015). *Multiple-Point Geostatistics: Stochastic Modeling with Training Images*. Chichester, UK: John Wiley & Sons. 364 pp.

Mariethoz, Gregoire and Bryce FJ Kelly (2011). "Modeling Complex Geological Structures with Elementary Training Images and Transform-Invariant Distances". In: *Water Resources Research* 47.7.

Mariethoz, Gregoire, Philippe Renard, and Julien Straubhaar (2010b). "The Direct Sampling Method to Perform Multiple-Point Geostatistical Simulations". In: *Water Resources Research* 46.11.

Mariethoz, Grégoire, Philippe Renard, and Jef Caers (2010c). "Bayesian Inverse Problem and Optimization with Iterative Spatial Resampling". In: *Water Resources Research* 46.11. ISSN: 1944-7973.

Marsily, G. de, J.P. Delhomme, A. Coudrain-Ribstein, and A.M. Lavenue (2000). "Four decades of inverse problems in hydrogeology". In: *Theory, Modeling, and Field Investigation in Hydrogeology: A Special Volume in Honor of Shlomo P. Neuman's 60th Birthday: Boulder, Colorado*. Ed. by D. Zhang and C.L. Winter. Vol. 348. The Geological Society of America, pp. 1–17.

Meerschman, Eef, Guillaume Pirot, Gregoire Mariethoz, Julien Straubhaar, Marc Van Meirvenne, and Philippe Renard (2013). "A Practical Guide to Performing Multiple-Point Statistical Simulations with the Direct Sampling Algorithm". In: *Computers & Geosciences* 52, pp. 307–324.

Mo, Shaoxing, Nicholas Zabaras, Xiaoqing Shi, and Jichun Wu (2019a). "Deep autoregressive neural networks for high-dimensional inverse problems in groundwater contaminant source identification". In: *Water Resources Research* 55.5, pp. 3856–3881.

Mo, Shaoxing, Yinhao Zhu, Nicholas Zabaras, Xiaoqing Shi, and Jichun Wu (2019b). "Deep Convolutional Encoder-Decoder Networks for Uncertainty Quantification of Dynamic Multiphase Flow in Heterogeneous Media". In: *Water Resources Research* 55.1, pp. 703–728.

Mosegaard, Klaus and Albert Tarantola (1995). "Monte Carlo sampling of solutions to inverse problems". In: *Journal of Geophysical Research: Solid Earth* 100.B7, pp. 12431–12447.

Mosser, Lukas, Olivier Dubrule, and Martin J Blunt (2017). "Reconstruction of three-dimensional porous media using generative adversarial neural networks". In: *Physical Review E* 96.4, p. 043309.

— (2018a). "Conditioning of three-dimensional generative adversarial networks for pore and reservoir-scale models". In: *arXiv preprint arXiv:1802.05622*.

— (2018b). "Stochastic reconstruction of an oolitic limestone by generative adversarial networks". In: *Transport in Porous Media* 125.1, pp. 81–103.

Mosser, Lukas, Olivier Dubrule, and Martin J. Blunt (2019). *DeepFlow: History Matching in the Space of Deep Generative Models*.

Al-Mudhafar, Watheq J. (2018). "Multiple-Point Geostatistical Lithofacies Simulation of Fluvial Sand-Rich Depositional Environment: A Case Study from Zubair Formation/South Rumaila Oil Field". In: *SPE Reservoir Evaluation & Engineering* 21.01, pp. 39–53.

Nair, Vinod and Geoffrey E Hinton (2010). "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814.

Neven, Alexis, Valentin Dall'Alba, Przemysław Juda, Julien Straubhaar, and Philippe Renard (2021). "Ice Volume and Basal Topography Estimation Using Geostatistical Methods and Ground-Penetrating Radar Measurements: Application to the Tsanfleuron and Scex Rouge Glaciers, Swiss Alps". In: *The Cryosphere* 15.11, pp. 5169–5186. ISSN: 1994-0416.

Ng, Hong-Wei, Viet Dung Nguyen, Vassilios Vonikakis, and Stefan Winkler (2015). "Deep learning for emotion recognition on small datasets using transfer learning". In: *Proceedings of the 2015 ACM on international conference on multimodal interaction*. ACM, pp. 443–449.

Oliver, Dean S and Yan Chen (2018). "Data Assimilation in Truncated Plurigaussian Models: Impact of the Truncation Map". In: *Mathematical Geosciences* 50.8, pp. 867–893.

Pan, Sinno Jialin and Qiang Yang (2009). "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10, pp. 1345–1359.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pérez, Cristian, Gregoire Mariethoz, and Julián M. Ortiz (2014). "Verifying the High-Order Consistency of Training Images with Data for Multiple-Point Geostatistics". In: *Computers & Geosciences* 70, pp. 190–205.

Pyrcz, Michael J. and C.V. Deutsch (2014). *Geostatistical Reservoir Modeling, Second Edition*. Oxford University Press.

Reed, Scott E, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee (2016). "Learning what and where to draw". In: *Advances in Neural Information Processing Systems*, pp. 217–225.

Rijsbergen, C. J. Van (1979). *Information Retrieval (2nd Ed.)* London: Butterworth-Heinemann.

Rodriguez, J. D., A. Perez, and J. A. Lozano (2010). "Sensitivity Analysis of K-Fold Cross Validation in Prediction Error Estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.3, pp. 569–575.

Rongier, Guillaume, Pauline Collon, Philippe Renard, Julien Straubhaar, and Judith Sausse (2016). "Comparing Connected Structures in Ensemble of Random Fields". In: *Advances in Water Resources* 96, pp. 145–169.

Selten, Reinhard (1998). "Axiomatic Characterization of the Quadratic Scoring Rule". In: *Experimental Economics* 1.1, pp. 43–61.

Shen, C., E. Laloy, A. Elshorbagy, A. Albert, J. Bales, F.-J. Chang, S. Ganguly, K.-L. Hsu, D. Kifer, Z. Fang, K. Fang, D. Li, X. Li, and W.-P. Tsai (2018). "HESS Opinions: Incubating Deep-Learning-Powered Hydrologic Science Advances as a Community". In: *Hydrology and Earth System Sciences* 22.11, pp. 5639–5656.

Stone, M. (1974). "Cross-Validatory Choice and Assessment of Statistical Predictions". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 36.2, pp. 111–133.

Straubhaar, J. (2019). *DeeSse user's guide*. Tech. rep. Centre for Hydrogeology and Geothermics (CHYN), University of Neuchâtel, Switzerland.

Straubhaar, Julien and Philippe Renard (May 2021). "Conditioning Multiple-Point Statistics Simulation to Inequality Data". In: *Earth and Space Science* 8.5. ISSN: 2333-5084, 2333-5084.

Straubhaar, Julien, Philippe Renard, and Tatiana Chugunova (2020). "Multiple-Point Statistics Using Multi-Resolution Images". In: *Stochastic Environmental Research and Risk Assessment* 34, pp. 251–273.

Strebelle, Sebastien (2002). "Conditional simulation of complex geological structures using multiple-point statistics". In: *Mathematical geology* 34.1, pp. 1–21.

Sun, Alexander Y (2018). "Discovering State-Parameter Mappings in Subsurface Models Using Generative Adversarial Networks". In: *Geophysical Research Letters* 45.20, pp. 11–137.

Tan, Xiaojin, Pejman Tahmasebi, and Jef Caers (2014). "Comparing Training-Image Based Algorithms Using an Analysis of Distance". In: *Mathematical Geosciences* 46.2, pp. 149–169.

Tarantola, Albert (2005). *Inverse Problem Theory and Methods for Model Parameter Estimation*. Philadelphia, PA: SIAM Society for Industrial and Applied Mathematics.

Tarantola, Albert and Bernard Valette (1982). "Inverse problems = quest for information". In: *Journal of Geophysics* 50, pp. 159–170.

Ter Braak, Cajo J. F. and Jasper A. Vrugt (Dec. 1, 2008). "Differential Evolution Markov Chain with Snooker Updater and Fewer Chains". In: *Statistics and Computing* 18.4, pp. 435–446. ISSN: 1573-1375.

Tripathy, Rohit K. and Ilias Bilionis (2018). "Deep UQ: Learning Deep Neural Network Surrogate Models for High Dimensional Uncertainty Quantification". In: *Journal of Computational Physics* 375, pp. 565–588. ISSN: 0021-9991.

Van Leeuwen, Matthijs, Chris B. M. te Stroet, Adrian P. Butler, and Jacob A. Tompkins (1998). "Stochastic Determination of Well Capture Zones". In: *Water Resources Research* 34.9, pp. 2215–2223.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric

Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17.3 (3), pp. 261–272. ISSN: 1548-7105.

Vrugt, Jasper A, CJF Ter Braak, CGH Diks, Bruce A Robinson, James M Hyman, and Dave Higdon (2009). "Accelerating Markov Chain Monte Carlo Simulation by Differential Evolution with Self-Adaptive Randomized Subspace Sampling". In: *International Journal of Nonlinear Sciences and Numerical Simulation* 10.3, pp. 273–290.

Xu, Yan, Tao Mo, Qiwei Feng, Peilin Zhong, Maode Lai, I Eric, and Chao Chang (2014). "Deep learning of feature representation with multiple instance learning for medical image analysis". In: *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp. 1626–1630.

Yang, Liu, Dongkun Zhang, and George Em Karniadakis (2018). "Physics-informed generative adversarial networks for stochastic differential equations". In: *arXiv preprint arXiv:1811.02033*.

Zahner, Tobias, Tobias Lochbühler, Grégoire Mariethoz, and Niklas Linde (2016). "Image Synthesis with Graph Cuts: A Fast Model Proposal Mechanism in Probabilistic Inversion". In: *Geophysical Journal International* 204.2, pp. 1179–1190. ISSN: 0956-540X.

Zhang, Ping (1993). "Model Selection Via Multifold Cross Validation". In: *Ann. Statist.* 21.1, pp. 299–313.

Zhou, Haiyan, J Jaime Gómez-Hernández, and Liangping Li (2014). "Inverse Methods in Hydrogeology: Evolution and Recent Trends". In: *Advances in Water Resources* 63, pp. 22–37.

Zhou, Haiyan, J. Jaime Gómez-Hernández, Harrie-Jan Hendricks Franssen, and Liangping Li (2011). "An Approach to Handling Non-Gaussianity of Parameters and State Variables in Ensemble Kalman Filtering". In: *Advances in Water Resources* 34.7, pp. 844–864. ISSN: 0309-1708.

Zhu, Yinhao and Nicholas Zabaras (2018). "Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification". In: *Journal of Computational Physics* 366, pp. 415–447.

Zhu, Yinhao, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris (2019). "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data". In: *Journal of Computational Physics* 394, pp. 56–81.

Zimmerman, DA, Ghislain De Marsily, Carol A Gotway, Melvin G Marietta, Carl L Axness, Richard L Beauheim, Rafael L Bras, Jesús Carrera, Gedeon Dagan, Paul B Davies, et al. (1998). "A Comparison of Seven Geostatistically Based Inverse Approaches to Estimate Transmissivities for Modeling Advective Transport by Groundwater Flow". In: *Water Resources Research* 34.6, pp. 1373–1413.