Journal of Petroleum Science and Engineering 195 (2020) 107763

Contents lists available at ScienceDirect

Journal of Petroleum Science and Engineering

journal homepage: www.elsevier.com/locate/petrol

4D seismic history matching: Assessing the use of a dictionary learning based sparse representation method

R.V. Soares [a,b,*], X. Luo [a], G. Evensen [a,c], T. Bhakta [a]

[a] NORCE Norwegian Research Centre, Norway
[b] University of Bergen, Norway
[c] Nansen Environmental and Remote Sensing Center (NERSC), Norway

ARTICLE INFO

Keywords:
Ensemble data assimilation
Iterative ensemble smoother
4D seismic history matching
Big data assimilation
Dictionary learning

ABSTRACT

It is possible to improve oil-reservoir simulation models by conditioning them on 4D seismic data. Computational issues may arise related to both storage and CPU time due to the size of the 4D seismic dataset. An approach to reducing the computational requirements is to use a sparse representation method, e.g., Dictionary Learning, to select only the main features of the 4D seismic data. However, the introduction of a sparse representation method incurs an additional computational cost. Alternatively, if one uses ensemble-based methods, it is possible to reduce storage and CPU time by projecting the full seismic dataset on a smaller subspace. This paper evaluates the potential of sparsely representing the seismic data. We compare two experiments, one where we condition on the full dataset projected on a smaller subspace, and one where we use Dictionary Learning to represent the data sparsely. We use Dictionary Learning both on the complete 4D seismic dataset and also on a denoised version of the data. We perform the data assimilation in a slightly different formulation of the Iterative Ensemble Smoother Regularized Levenberg–Marquardt together with correlation-based adaptive localization. We apply these methods to the Brugge benchmark case. Experiment results show that sparse representation methods lead to a final ensemble that is closer to the reference solution, and denoising the seismic data before applying the sparse representation allows us to capture the 4D effect better. Thus, using a sparse representation method in 4D-seismic history matching leads to improved results compared to what we obtain when conditioning the models on the projected 4D seismic dataset.

1. Introduction

Data assimilation, also known as history matching in reservoir engineering, denotes the process of conditioning reservoir models on dynamical observations from a reservoir, e.g., well rates and 4D seismic data, to update uncertain reservoir model parameters, e.g., porosity and permeability. Historically, most of the data assimilation processes have been mainly using production data from the wells because of its availability and easiness of gather. There is a vast list in the literature that successfully performed history matching in various benchmark and real field cases. See, e.g., Emerick and Reynolds (2011), Oliver and Chen (2011), Chen and Oliver (2014), Goda and Sato (2014), Maschio and Schiozer (2016), and Soares et al. (2018), for some history-matching experiments.

Even though it might be possible to achieve a good representation of the reservoir using only well data, the inclusion of different data, such as seismic, can help to understand the flow dynamics in the oil reservoir better. Johnston (2013) mentioned several important points for practical applications of 4D seismic, including rock physics models, seismic processing, data integration among others. Also, the author pointed out that seismic data are much denser in space than in time. Therefore, 4D seismic data will complement the information we have from rate data in the history-matching process. Several authors have proved the benefits of using 4D seismic data in history matching, both in benchmark and real cases (Emerick, 2016; Roggero et al., 2012; Fahimuddin et al., 2010; Obidegwu et al., 2017; Leeuwenburgh and Arts, 2014; Helgerud et al., 2011; Almeida et al., 2020).

There are several challenges when considering 4D seismic data in data assimilation. As commercial simulators usually provide only well rate and pressure and saturation maps, one needs to bring simulator response and seismic into the same domain. In other words, one has to transform acquired seismic data into pressure and saturation maps or bring them into another domain, such as acoustic impedance or amplitude-versus-angle (AVA). Besides, seismic data comprise huge datasets and, therefore, issues regarding computational storage and CPU time may arise. To deal with the big-data issue,

* Corresponding author at: NORCE Norwegian Research Centre, Norway.
E-mail address: riso@norceresearch.no (R.V. Soares).

https://doi.org/10.1016/j.petrol.2020.107763
Received 25 February 2020; Received in revised form 5 August 2020; Accepted 7 August 2020
Available online 14 August 2020

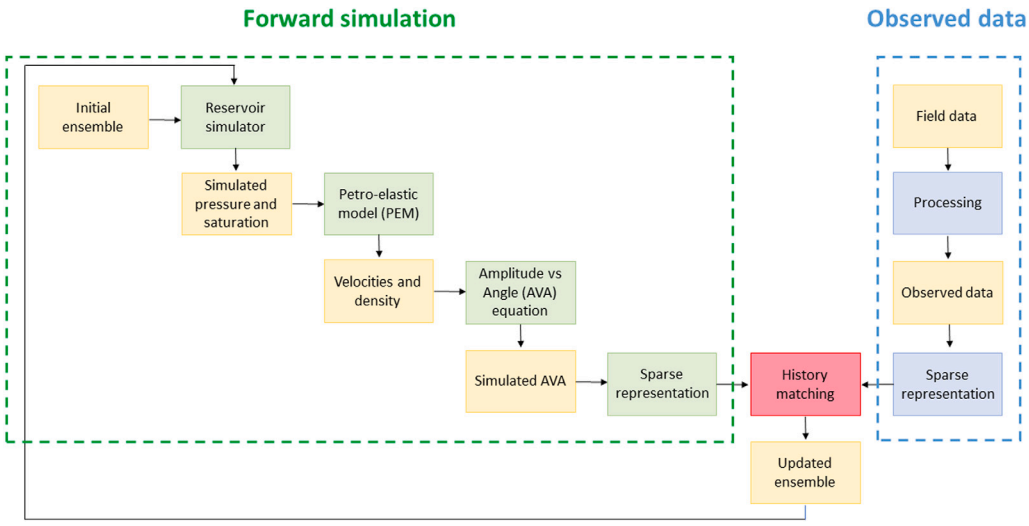

Fig. 1. Flowchart of the use of 4D seismic data in history matching. The area within the green dashed rectangle represents the forward simulation of seismic data, while the area within the blue dashed rectangle represents the observed data from the field. The yellow boxes represent data, and the blue, green, and red boxes represent the required processing steps. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

authors such as Luo et al. (2017), Liu and Grana (2020) and Soares et al. (2019) suggested different methodologies to represent the seismic data sparsely.

Fig. 1 depicts a flowchart concerning the use of 4D seismic data in history matching (Luo et al., 2018), where the authors consider AVA as the seismic data. The area covered by the green dashed border lines (referred to as the green part hereafter) represents the simulated data (forward simulation of seismic), while the area enclosed by the blue dashed border lines (named blue part hereafter) represents the observed data from the field. The yellow boxes represent data, and the blue, green, and red boxes represent the necessary steps to deal with this kind of data.

The green part describes the forward seismic simulation workflow that generates simulated seismic attributes (AVA in this case) from an ensemble of reservoir models. The boxes in the blue part convert seismic from raw waveforms to AVA data. This conversion procedure involves seismic processing, which is outside the scope of the current work. Thus, we assume that the observed AVA attributes in the blue part are readily available. Note that one can also assimilate other types of seismic attributes, such as Acoustic Impedance (AI) following a similar workflow (Fahimuddin et al., 2010; Roggero et al., 2012; Alfonzo and Oliver, 2019), or other parameters that can be inferred from seismic, such as fluid fronts (Leeuwenburgh and Arts, 2014; Kretz et al., 2004).

As mentioned previously, the 4D seismic dataset is very big. Therefore, the flowchart considers a method capable of sparsely representing the seismic signal while retaining the main characteristics of the data. For this purpose, Luo et al. (2018) and Lorentzen et al. (2019) used the Discrete Wavelet Transform (DWT) to represent the 4D seismic dataset sparsely. Through DWT, they separated the data into low and high-frequency sub-bands. Then, they estimated the noise of the wavelet coefficients and applied a threshold value to retain only the leading wavelet coefficients. Although they reduced the number of retained coefficients substantially, it was necessary to define an empirical parameter to achieve a trade-off between the number of coefficients and the retained characteristics, which we will illustrate below (cf. Fig. 10).

Liu and Grana (2020) proposed to learn a sparse representation of 4D seismic data through a deep convolutional autoencoder. The authors performed experiments in a 2D and 3D model and concluded that the method improved the data-assimilation performance. Nevertheless, one might need a more powerful tool, such as GPU, to obtain the results in a reasonable computational time. For the 3D model, for instance, they reduced the original dataset to 0.20% of its original size, and it took about 30 min to train the model with dimension $64 \times 64 \times 64$ in two different time surveys. Besides, Canchumuni et al. (2019) reported that an extension of the deep learning method from 2D to 3D case studies might incur substantially more computational costs.

Soares et al. (2019) investigated the use of a Dictionary Learning (DL) method for sparse representation of seismic data in a 3D case study with dimension $139 \times 48 \times 176$. The method retained the main characteristics of the 4D seismic by using a set of non-zero coefficients whose number was only about 0.25% of the original data size, and the related computational time was about 6.7 s.

The end products of the sparse representation procedure (applied to observed seismic data) will be the obtained non-zero coefficients. These non-zero coefficients will then be taken as the observations in 4D seismic history matching problems, as indicated in the blue part of Fig. 1. By doing so, a practical benefit is that the effective data size can be substantially smaller than the size of the original 4D seismic dataset, which will then help to mitigate the issues of computational memory and time during history matching.

Iterative ensemble-based methods are among state of the art for history matching, as they are relatively easy to implement, and have the capacity of dealing with large scale problems (Evensen, 2009a). In ensemble-based methods, one can choose to project big 4D seismic data onto a subspace which is related to the ensemble of reservoir models, and whose dimension is no more than the ensemble size. These projected data then serve as the effective observations (Lorentzen et al., 2019; Luo et al., 2019; Chen and Oliver, 2017). This approach is an alternative way to handle the issue of big data in history matching problems, yet without using any sparse representation method.

Therefore, there are two possible ways to deal with the big data issue: through sparse representations methods and through the projection of the observed data into a subspace related to the ensemble models. Note that in both approaches, one faces information loss in the dataset. However, the use of a sparse data representation introduces additional computational time. Hence, compared to the projection-based approach, is it still beneficial to use a sparse representation method for data-size reduction? To answer this question, we will examine the data assimilation performance in two sets of experiments: one employing the projection-based approach and the other adopting a Dictionary Learning based sparse presentation method. Furthermore, we also investigate an alternative way to apply the Dictionary Learning based sparse representation to obtain further improved assimilation performance. To perform history matching, we modify the Iterative

Ensemble Smoother Regularized Levenberg–Marquardt (IES-RML) developed by Luo et al. (2015) in a way that there is no need to perform the truncated singular value decomposition during data assimilation.

In the following, we start by explaining each component of the flowchart shown in Fig. 1. We divide this part into three sections. First, we explain the individual steps in the forward seismic simulation workflow. Then, we discuss the history matching part, where we focus on ensemble-based methods and we provide more details regarding the need of sparse representation methods. Afterwards, we introduce one of the Dictionary Learning methods, called the K-SVD algorithm, which is responsible for sparsely representing the data. Subsequently, we provide information about the case study (the Brugge benchmark case), methodology, and application. Finally, we show and discuss the results and draw conclusions.

As the different sections present unrelated topics, we prefer to keep the traditional notation for each topic and we introduce a list of symbols and variables at the beginning of each section.

2. Forward simulation of 4D seismic data

Nomenclature

μ	Shear modulus
ν	Poisson's ratio
ϕ	Porosity
ρ	Density
C_p	Average number of contacts per sphere
K	Bulk modulus
n	Degree of root
P	Pressure
S	Saturation
V	Wave velocity
Subscripts	
c	Critical
d	Rock grain porosity values lower than ϕ_c
eff	Effective
f	Saturation effect on the presence of oil and water
HM	Hertz–Mindlin model — dry rock at critical porosity
m	Mineral
o	Oil
P	Compressional wave
s	Rock grain
S	Shear wave
sat	Saturation effect
w	Water

In this section, we provide information about the green boxes in Fig. 1, except the Sparse Representation, which will be given in Section 4 (Observed Data — Sparse Representation: Dictionary Learning Through the K-SVD Algorithm).

Once one generates the initial ensemble containing the uncertain model variables, such as porosity, permeability, and fault multipliers, the next step is to run the simulation models to obtain simulated pressure and saturation profiles, which are among the important quantities to calculate the simulated seismic data.

2.1. Petro-elastic model (PEM)

Petro-elastic models convert pressure and saturation data into elastic attributes, such as acoustic impedance (or equivalently, wave velocities and densities), for seismic interpretation or inversion. PEM is dependent on the type of reservoir formation and fluids.

One of the most commonly used PEMs is the soft-sand model (Mavko et al., 2009), in which the first step is to calculate the dry bulk modulus (K_{HM}) and the shear modulus (μ_{HM}) through the Hertz–Mindlin model (Mindlin, 1949) as in

$$K_{\mathrm{HM}} = \sqrt[n]{\frac{C_p^2(1-\phi_c)^2\mu_s^2}{18\pi^2(1-\nu_s)^2}P_{\mathrm{eff}}}, \tag{1}$$

and

$$\mu_{\mathrm{HM}} = \frac{5-4\nu_s}{5(2-\nu_s)}\sqrt[n]{\frac{3C_p^2(1-\phi_c)^2\mu_s^2}{2\pi^2(1-\nu_s)^2}P_{\mathrm{eff}}}. \tag{2}$$

In Eqs. (1) and (2), n is the degree of root, C_p is the average number of contacts per sphere, ϕ_c is the critical porosity, μ_s is the grain shear modulus, ν_s is the Poisson's ratio, and P_{eff} is the effective stress, i.e., the lithostatic pressure minus pore pressure. In this work, n is set to 3, C_p is set to 9 and ϕ_c to 36% (the maximum porosity value). Physically, one can interpret K_{HM} and μ_{HM} as the dry rock resistance to normal and shear stress, respectively. In addition, one can calculate Poisson's ratio as in

$$\nu_s = \frac{3K_s-\mu_s}{6K_s-\mu_s}. \tag{3}$$

K_s and μ_s are bulk modulus and the shear modulus of the rock grain, respectively.

The modified Lower Hashin–Shtrikman (MLHS) (Hashin and Shtrikman, 1963; Mavko et al., 2009) calculates the effective dry bulk modulus (K_d) and the effective shear modulus (μ_d) for porosity values (ϕ) lower than ϕ_c

$$K_d = \left(\frac{\frac{\phi}{\phi_c}}{K_{\mathrm{HM}}+\frac{4}{3}\mu_{\mathrm{HM}}} + \frac{\frac{1-\phi}{\phi_c}}{K_s+\frac{4}{3}\mu_{\mathrm{HM}}}\right)^{-1} - \frac{4}{3}\mu_{\mathrm{HM}}, \tag{4}$$

and

$$\mu_d = \left(\frac{\frac{\phi}{\phi_c}}{\mu_{\mathrm{HM}}+\frac{\mu_{\mathrm{HM}}}{6}Z} + \frac{\frac{1-\phi}{\phi_c}}{\mu_s+\frac{\mu_{\mathrm{HM}}}{6}Z}\right)^{-1} - \frac{\mu_{\mathrm{HM}}}{6}Z, \tag{5}$$

with

$$Z = \frac{9K_{\mathrm{HM}}+8\mu_{\mathrm{HM}}}{K_{\mathrm{HM}}+2\mu_{\mathrm{HM}}}. \tag{6}$$

The next step is known as the Gassmann model (Hashin and Shtrikman, 1951), which generates the saturated bulk modulus and shear modulus (K_{sat} and μ_{sat}, respectively) by including the saturation effect (Eqs. (7) and (8)).

$$K_{sat} = K_d + \frac{\left(1-\frac{K_d}{K_s}\right)^2}{\frac{\phi}{K_f}+\frac{1-\phi}{K_s}-\frac{K_d}{K_s^2}}, \tag{7}$$

$$\mu_{sat} = \mu_d, \tag{8}$$

and K_f takes into consideration the presence of both water and oil

$$K_f = \left(\frac{S_w}{K_w}+\frac{S_o}{K_o}\right)^{-1}. \tag{9}$$

Here K_o and K_w are the bulk modulus of the oil and water, respectively, while S_o and S_w are the oil and water saturation. Subsequently, one calculates the saturated density (ρ_{sat}), P-wave and S-wave velocities (V_P and V_S), as in Mavko et al. (2009)

$$\rho_{sat} = (1-\phi)\rho_m + \phi S_w\rho_w + \phi S_o\rho_o, \tag{10}$$

$$V_p = \sqrt{\frac{K_{sat}+\frac{4}{3}\mu_{sat}}{\rho_{sat}}}, \tag{11}$$

and

$$V_s = \sqrt{\frac{\mu_{sat}}{\rho_{sat}}}, \tag{12}$$

where ρ_m, ρ_o, ρ_w and, ρ_{sat} are mineral density, oil density, water density, and saturated rock density, respectively. Note that one can obtain wave impedance by multiplying ρ_{sat} by V_p or V_s.

2.2. AVA equation

To calculate the AVA, one first needs to define the reflection coefficients at each interface between two adjacent layers by using the Zoeppritz equation (Avseth et al., 2010). Then, one should compute the travel time using V_p and the thickness of the gridblocks. Finally, the AVA data can be obtained by convolving the reflectivity series with a Ricker wavelet. Here, a Ricker wavelet of 45 Hz is considered. Note that different incident angles can be considered while generating the AVA data. The readers are referred to Luo et al. (2017) for more information about how to generate the AVA data.

In seismic history-matching problems, acoustic impedance can also be adopted, see, for example, Lorentzen et al. (2019), Emerick (2016), and Gosselin et al. (2003). However, AVA has the advantage of avoiding the inclusion of biases or errors during the seismic inversion process (Luo et al., 2018) and AVA can help to identify different situations in the reservoir.

3. History matching: Ensemble-based methods

Nomenclature	
α	Regularization/Weight term
β	Scalar parameter responsible to increase/decrease α
$\boldsymbol{\delta}$	Measurement error
ϵ	Noise in the correlation calculated in the localization
ρ	Correlation between the model variables in the ensemble and the simulated data
σ	Standard deviation
$\boldsymbol{\Sigma}$	Singular values matrix
$\hat{\boldsymbol{\Sigma}}$	Truncated singular values matrix
θ	Threshold value in the automatic and adaptive correlation-based localization
ζ	Average data mismatch
$\mathbf{A}$	Ensemble anomalies
c	Entry of the localization matrix
$\mathbf{C}$	Localization matrix
$\mathbf{C}_\mathrm{d}$	Measurement error covariance matrix
$\mathbf{C}_\mathrm{x}$	Uncertain model parameter error covariance matrix
$\mathbf{d}$	Observed data
$\mathbf{d}^\mathrm{sim}$	Simulated data
$\mathbf{D}$	Ensemble matrix of observation residual
$\widetilde{\mathbf{D}}$	Rotated $\mathbf{D}$
$\mathbf{g}$	Reservoir simulator
$\mathbf{K}$	Kalman gain
N_d	Total number of observed data
N_e	Total number of realizations in the ensemble
N_i	Total number of iterations
N_sv	Total number of retained singular values
N_x	Total number of uncertain variables
$p(\cdot)$	Prior distribution
$p(\cdot\|\cdot)$	Likelihood or posterior probability density function given a parameter
$\mathbf{S}$	Simulated data anomalies
$\widetilde{\mathbf{S}}$	Rotated $\mathbf{S}$
$\mathbf{U}$	Left singular vectors matrix
$\hat{\mathbf{U}}$	Truncated left singular vectors matrix
$\mathbf{V}$	Right singular vectors matrix
$\hat{\mathbf{V}}$	Truncated right singular vectors matrix
$\mathbf{x}$	Uncertain model parameter
$\mathbf{X}$	Ensemble matrix of uncertain model variables
z	Dummy variable defined in the correlation-based adaptive localization
Subscripts	
c	Common point
eff	Effective
j	Realization index
Superscripts	
0	Prior
i	Iteration index
true	True
T	Transpose

We will now introduce the ensemble method used for conditioning the model on the observations (red box in Fig. 1). Model conditioning or history matching is an essential component in the development and management of petroleum reservoirs. By using the information about the dynamical properties of the reservoir contained in observed data, such as well rates (oil, water, and gas) and seismic, one can constrain reservoir models and update the uncertain model parameters, such as permeability and porosity fields. The Ensemble Kalman Filter (EnKF) by Evensen (1994), initially developed for sequential data assimilation, provides an efficient means for model conditioning. Nævdal et al. (2002) introduced the EnKF for use in the petroleum industry, and since then, ensemble-based methods have attracted a lot of attention within the industry. Notably, the introduction of the Ensemble Smoother (van Leeuwen and Evensen, 1996) (which is an EnKF that uses all data simultaneously to update all parameters in one step) for petroleum applications by Skjervheim et al. (2011) triggered further development of effective iterative forms of the Ensemble Smoother.

3.1. History-matching problem

To formulate the history-matching problem, we denote the uncertain model parameters as $\mathbf{x}$, the (potentially imperfect) forward operator which transforms the model variables into the simulated data $\mathbf{d}^\mathrm{sim}$ as $\mathbf{g}$, such that

$$\mathbf{d}^\mathrm{sim} = \mathbf{g}(\mathbf{x}). \tag{13}$$

Besides, we have one set of observed data $\mathbf{d}$, which are generated through the following observation system

$$\mathbf{d} = \mathbf{g}^\mathrm{true}(\mathbf{x}^\mathrm{true}) + \boldsymbol{\delta}. \tag{14}$$

Here, $\mathbf{x}^\mathrm{true}$ stands for the ground-truth model, and $\boldsymbol{\delta}$ represents measurement errors that are normally distributed random errors with zero mean and variance represented by the measurement error-covariance matrix $\mathbf{C}_\mathrm{d}$, e.g., $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_\mathrm{d})$. The true (but unknown) forward simulator $\mathbf{g}^\mathrm{true}$ may be different from the actual forward operator $\mathbf{g}$ in use, hence model errors can arise. In this work, however, we do not consider model errors as in Luo and Bhakta (2020), Chen and Oliver (2017), and Emerick (2016), for instance. More information on how to deal with model errors can be found in Evensen (2019) and Luo (2019).

Since $\mathbf{x}$ is uncertain, history matching is an inverse problem where one solves for a $\mathbf{x}$ that results in a model prediction $\mathbf{g}(\mathbf{x})$ that is close to the observed data $\mathbf{d}$. Furthermore, the number of parameters is normally much larger than the independent degrees of freedom in the observations. In this case, estimating model variables $\mathbf{x}$ based on the observations $\mathbf{d}$ and the forward simulator $\mathbf{g}$ defines a high-dimensional and under-determined inverse problem.

Through Bayes' theorem, it is possible to define the posterior probability density function of $\mathbf{x}$ given the observed data

$$p(\mathbf{x}|\mathbf{d}) \propto p(\mathbf{d}|\mathbf{x})p(\mathbf{x}), \tag{15}$$

where $p(\mathbf{x})$ is the prior distribution of the model parameters and $p(\mathbf{d}|\mathbf{x})$ is the likelihood. By assuming that both the prior and likelihood are Gaussian distributed and considering an ensemble of multiple realizations (Evensen, 2009a), it is possible to sample the posterior

distribution $p(\mathbf{x}|\mathbf{d})$ by the minimization of an ensemble of cost functions associated for each realization j

$$\underset{\{\mathbf{x}_j\}}{\operatorname{argmin}}\left\{ \left(\mathbf{g}(\mathbf{x}_j)-\mathbf{d}_j\right)^{\mathrm{T}}\mathbf{C}_{\mathrm{d}}^{-1}\left(\mathbf{g}(\mathbf{x}_j)-\mathbf{d}_j\right)+\left(\mathbf{x}_j-\mathbf{x}_j^0\right)^{\mathrm{T}}\mathbf{C}_{\mathrm{x}}^{-1}\left(\mathbf{x}_j-\mathbf{x}_j^0\right)\right\}. \quad (16)$$

Here, $\mathbf{x}^0$ is the prior estimate of the model parameters and $\mathbf{C}_{\mathrm{x}}$ is the error covariance matrix of the uncertain model parameters (or the prior covariance matrix for the model parameters). Note that one needs to sample a prior ensemble of models $\mathbf{x}_j^0$ from $\mathcal{N}(\mathbf{x}^0,\mathbf{C}_{\mathrm{x}})$ and an ensemble of measurements $\mathbf{d}_j$ from $\mathcal{N}(\mathbf{d},\mathbf{C}_{\mathrm{d}})$.

For the case where $\mathbf{g}$ is linear, the minimization of the Eq. (16) will sample the correct posterior distribution for an infinite ensemble size, N_{e}, Evensen (2009a). However, if $\mathbf{g}$ is non-linear, as in the case of reservoir applications, the minimization of the Eq. (16) is an approximation.

By minimizing Eq. (16), one can obtain the following formula to update the uncertain model variables

$$\mathbf{x}_j=\mathbf{x}_j^0+\mathbf{A}\mathbf{S}^{\mathrm{T}}\left(\mathbf{S}\mathbf{S}^{\mathrm{T}}+\mathbf{C}_{\mathrm{d}}\right)^{-1}\left(\mathbf{d}_j-\mathbf{g}(\mathbf{x}_j^0)\right), \quad (17)$$

where

$$\mathbf{A}=\frac{1}{\sqrt{N_{\mathrm{e}}-1}}\left(\mathbf{x}_1^0-\mathbf{x}_c^0,\ldots,\mathbf{x}_{N_{\mathrm{e}}}^0-\mathbf{x}_c^0\right);\ \mathbf{x}_c^0=\frac{1}{N_{\mathrm{e}}}\sum_{j=1}^{N_{\mathrm{e}}}\mathbf{x}_j^0, \quad (18)$$

$$\mathbf{S}=\frac{1}{\sqrt{N_{\mathrm{e}}-1}}\left(\mathbf{g}(\mathbf{x}_1^0)-\bar{\mathbf{d}},\ldots,\mathbf{g}(\mathbf{x}_{N_{\mathrm{e}}}^0)-\bar{\mathbf{d}}\right);\ \bar{\mathbf{d}}=\frac{1}{N_{\mathrm{e}}}\sum_{j=1}^{N_{\mathrm{e}}}\mathbf{g}(\mathbf{x}_j^0), \quad (19)$$

and N_{e} is the total number of realizations in the ensemble. Note that $\mathbf{C}_{\mathrm{x}}=\mathbf{A}(\mathbf{A})^{\mathrm{T}}$. Eq. (17) is known as the analysis equation for the Ensemble Smoother (van Leeuwen and Evensen, 1996). For more information about the minimization process and ensemble-based methods in general, the readers are referred to Evensen (2009a) and Aanonsen et al. (2009).

To reduce the impact of the approximation, one can use iterative forms of the ensemble smoother as they generate better approximate solutions than the non-iterative form (Evensen, 2018). Among recent iterative ensemble-smoothers, the most popular are the EnRML method by Chen and Oliver (2012, 2013) and the Ensemble Smoother with Multiple Data Assimilation (ESMDA) by Emerick and Reynolds (2013). A recent reformulation of the EnRML by Evensen et al. (2019) and Raanes et al. (2019) is a more efficient alternative than the original EnRML method. In this work, we employ the Iterative Ensemble Smoother Regularized Levenberg–Marquardt (IES-RML) by Luo et al. (2015). Even though the different iterative smoothers are similar to each other, they solve slightly different problems, and it is difficult to choose the best one since the results may be case dependent (Evensen, 2018). In the next section, we provide more information about the IES-RLM, which we use in this work.

3.2. IES-RLM

Luo et al. (2015) were more interested in the average change in data mismatch between iterations. Hence, they proposed the following sequence of minimization problems for each realization

$$\underset{\left\{\mathbf{x}_j^{i+1}\right\}}{\operatorname{argmin}}\frac{1}{N_{\mathrm{e}}}\sum_{j=1}^{N_{\mathrm{e}}}\left\{\left(\mathbf{d}_j-\mathbf{g}\left(\mathbf{x}_j^{i+1}\right)\right)^{\mathrm{T}}\mathbf{C}_{\mathrm{d}}^{-1}\left(\mathbf{d}_j-\mathbf{g}\left(\mathbf{x}_j^{i+1}\right)\right)+\alpha^i\left(\mathbf{x}_j^{i+1}-\mathbf{x}_j^i\right)^{\mathrm{T}}\mathbf{C}_{\mathrm{x}}^{-1}\left(\mathbf{x}_j^{i+1}-\mathbf{x}_j^i\right)\right\}, \quad (20)$$

where i is the iteration number ($i=0,1,\ldots,N_{\mathrm{i}}$). The first term accounts for the difference between the simulated and observed data as in Eq. (16), while the second term penalizes the update increments in each iteration regarding the prior ensemble. In this formulation, α^i is a positive scalar responsible for assigning a weight to the second term.

The minimization of the Eq. (20) gives us the following formula to update the uncertain model variables for the IES-RML method

$$\mathbf{x}_j^{i+1}=\mathbf{x}_j^i+\mathbf{A}^i\left(\mathbf{S}^i\right)^{\mathrm{T}}\left(\mathbf{S}^i\left(\mathbf{S}^i\right)^{\mathrm{T}}+\alpha^i\mathbf{C}_{\mathrm{d}}\right)^{-1}\left(\mathbf{d}_j-\mathbf{g}\left(\mathbf{x}_j^i\right)\right). \quad (21)$$

Here, $\mathbf{A}^i$ is similar as in Eq. (18), with the exception that now one calculates them at each iteration, α^i is the regularization coefficient which changes over the iteration steps, to be explained later, and $\mathbf{S}^i$ is defined as

$$\mathbf{S}^i=\frac{1}{\sqrt{N_{\mathrm{e}}-1}}\left(\mathbf{g}\left(\mathbf{x}_1^i\right)-\mathbf{g}\left(\mathbf{x}_c^i\right),\ldots,\mathbf{g}\left(\mathbf{x}_{N_{\mathrm{e}}}^i\right)-\mathbf{g}\left(\mathbf{x}_c^i\right)\right). \quad (22)$$

In this algorithm, one needs to choose certain stopping criteria for the number of iterations. According to Luo et al. (2015), such criteria should be based on the following:

1. If the data mismatch becomes smaller than the number of observed data points (N_{d}) times a factor (1 in this study).
2. A maximum number of iterations (20 in this study).
3. The relative change between data mismatch in two consecutive iterations is small (less than 0.01% in this study).

The average data mismatch used in criterion 1 and 3 is defined as

$$\zeta=\frac{1}{N_{\mathrm{e}}}\sum_{j=1}^{N_{\mathrm{e}}}\left(\mathbf{d}_j-\mathbf{g}\left(\mathbf{x}_j\right)\right)^{\mathrm{T}}\mathbf{C}_{\mathrm{d}}^{-1}\left(\mathbf{d}_j-\mathbf{g}\left(\mathbf{x}_j\right)\right). \quad (23)$$

We now define the ensemble matrix

$$\mathbf{X}^i=\left(\mathbf{x}_1^i,\mathbf{x}_2^i,\ldots,\mathbf{x}_{N_{\mathrm{e}}}^i\right), \quad (24)$$

and additionally one can define the ensemble matrix of observation residuals as the observed data minus the simulated data as in

$$\mathbf{D}^i=\left(\mathbf{d}_1-\mathbf{g}\left(\mathbf{x}_1^i\right),\ldots,\mathbf{d}_{N_{\mathrm{e}}}-\mathbf{g}\left(\mathbf{x}_{N_{\mathrm{e}}}^i\right)\right). \quad (25)$$

Besides,assuming that we have access to a symmetric square root $\mathbf{C}_{\mathrm{d}}^{\frac{1}{2}}$ of $\mathbf{C}_{\mathrm{d}}$ such that $\mathbf{C}_{\mathrm{d}}=\mathbf{C}_{\mathrm{d}}^{\frac{1}{2}}\left(\mathbf{C}_{\mathrm{d}}^{\frac{1}{2}}\right)^{\mathrm{T}}$, and considering the following rotated operators

$$\widetilde{\mathbf{S}}^i=\mathbf{C}_{\mathrm{d}}^{-\frac{1}{2}}\mathbf{S}^i, \quad (26)$$

$$\widetilde{\mathbf{D}}^i=\mathbf{C}_{\mathrm{d}}^{-\frac{1}{2}}\mathbf{D}^i, \quad (27)$$

one can write the analysis equation as

$$\mathbf{X}^{i+1}=\mathbf{X}^i+\mathbf{A}^i\left(\widetilde{\mathbf{S}}^i\right)^{\mathrm{T}}\left(\widetilde{\mathbf{S}}^i\left(\widetilde{\mathbf{S}}^i\right)^{\mathrm{T}}+\alpha^i\mathbf{I}\right)^{-1}\widetilde{\mathbf{D}}^i, \quad (28)$$

Note that in many practical cases, $\mathbf{C}_{\mathrm{d}}$ is assumed to be diagonal (uncorrelated measurement errors). Hence, Eqs. (27) and (26) can be viewed as a re-scaling.

Nevertheless, the matrix product to be inverted, $\left(\widetilde{\mathbf{S}}^i\left(\widetilde{\mathbf{S}}^i\right)^{\mathrm{T}}\right)$, has dimension $N_{\mathrm{d}}\times N_{\mathrm{d}}$, which can be very big (especially when using 4D seismic data). Therefore, a common procedure in the literature to obtain a more numerically stable algorithm is to apply the Truncated Singular Value Decomposition (TSVD) to the matrix $\widetilde{\mathbf{S}}^i$. In this regard, Evensen (2009a) suggested to keep the leading singular values that add up to between 90% and 99.9% of the total sum of squared singular values. Suppose that, through the TSVD, one obtains

$$\widetilde{\mathbf{S}}^i\approx\hat{\mathbf{U}}^i\hat{\mathbf{\Sigma}}^i(\hat{\mathbf{V}}^i)^{\mathrm{T}}, \quad (29)$$

where the matrices with respect to singular vectors or values, $\hat{\mathbf{U}}^i$, $\hat{\mathbf{\Sigma}}^i$, and $(\hat{\mathbf{V}}^i)^T$, have the dimensions of $N_{\mathrm{d}}\times N_{\mathrm{sv}}$, $N_{\mathrm{sv}}\times N_{\mathrm{sv}}$, and $N_{\mathrm{sv}}\times N_{\mathrm{e}}$, respectively. N_{sv} is the number of kept leading singular values.

Inserting Eq. (29) into Eq. (28) and applying some algebra, one has

$$\mathbf{X}^{i+1}=\mathbf{X}^i+\mathbf{A}^i\hat{\mathbf{V}}^i(\hat{\mathbf{\Sigma}}^i)^{\mathrm{T}}\left(\hat{\mathbf{\Sigma}}^i(\hat{\mathbf{\Sigma}}^i)^{\mathrm{T}}+\alpha^i\mathbf{I}\right)^{-1}(\hat{\mathbf{U}}^i)^{\mathrm{T}}\widetilde{\mathbf{D}}^i. \quad (30)$$

Several authors, such as Luo et al. (2015), Luo and Bhakta (2020), Lorentzen et al. (2019) and Chen and Oliver (2017), used this formulation. Nevertheless, if we consider the Woodbury identity (see, e.g., Evensen et al., 2019)

$$\mathbf{B}^{\mathrm{T}}(\mathbf{B}\mathbf{B}^{\mathrm{T}}+\mathbf{R})^{-1}=(\mathbf{I}+\mathbf{B}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{R}^{-1}, \tag{31}$$

it is possible to rearrange Eq. (28) into the following form

$$\mathbf{X}^{i+1}=\mathbf{X}^{i}+\mathbf{A}^{i}\left((\widetilde{\mathbf{S}}^{i})^{\mathrm{T}}\widetilde{\mathbf{S}}^{i}+\alpha^{i}\mathbf{I}\right)^{-1}(\widetilde{\mathbf{S}}^{i})^{\mathrm{T}}\widetilde{\mathbf{D}}^{i}. \tag{32}$$

The nice property of the Eq. (32) is that the inversion is of a matrix of dimension the ensemble size $N_e \times N_e$ rather than the measurement dimension ($N_d \times N_d$) in Eq. (28). Therefore, one can avoid the application of the TSVD.

The final matrix product in Eq. (32) $\left((\widetilde{\mathbf{S}}^{i})^{\mathrm{T}}\widetilde{\mathbf{D}}^{i}\right)$ is effectively a projection of the observation residuals onto the ensemble space spanned by $\widetilde{\mathbf{S}}^{i}$. Thus, by solving the update equation with Eq. (32), one can only exploit information from the measurements that can be represented by the ensemble. In addition, the matrix multiplication $\left((\widetilde{\mathbf{S}}^{i})^{\mathrm{T}}\widetilde{\mathbf{D}}^{i}\right)$ helps to handle the big data assimilation problem, as it produces a lower-dimensional representation of the observation residuals.

Given Eq. (32), the regularization parameter α^i is determined following the rule below (Luo et al., 2015, 2019)

$$\alpha^{i}=\beta^{i}\times\mathrm{trace}\left((\widetilde{\mathbf{S}}^{i})^{\mathrm{T}}\widetilde{\mathbf{S}}^{i}\right)/N_{e}, \tag{33}$$

where β^i is a scalar that should decrease by a certain rate if the average data mismatch is reduced while increasing by a certain rate instead if the average data mismatch increases, and trace is an operator that calculates the trace of a matrix. Note that depending on the formulation, the calculation of α^i will change, as it depends on the matrix that is being summed $\left((\widetilde{\mathbf{S}}^{i})^{\mathrm{T}}\widetilde{\mathbf{S}}^{i}\right)$, in this case.

Nevertheless, the limited number of realizations in the ensemble and a possible big dataset in the observed data can make the problem overdetermined, which would cause a strong reduction of the ensemble variability. An approach much used to deal with this issue is the application of a localization technique.

3.3. Localization

The traditional view is that localization is required to mitigate the effect of spurious correlation from remote observations, which leads to variance reduction in the ensemble (Aanonsen et al., 2009; Emerick and Reynolds, 2011; Chen and Oliver, 2017; Sakov and Bertino, 2011; Devegowda et al., 2007; Luo and Bhakta, 2020; Soares et al., 2018). Localization is described by the matrix $\mathbf{C}$, which represents the relationship between each observed data and the uncertain model variables. Hence, one can use the localization matrix $\mathbf{C}$ to specify which data should be used to update which set of uncertain parameters.

To conduct localization, it is better to rearrange Eq. (32) as in

$$\mathbf{X}^{i+1}=\mathbf{X}^{i}+\mathbf{K}^{i}\widetilde{\mathbf{D}}^{i}, \tag{34}$$

where we have introduced the so-called Kalman gain matrix $\mathbf{K}^i \in \Re^{N_x \times N_d}$ that can take different forms using either one of the Eqs. (28) or (32), e.g.,

$$\mathbf{K}^{i}=\mathbf{A}^{i}\left((\widetilde{\mathbf{S}}^{i})^{\mathrm{T}}\widetilde{\mathbf{S}}^{i}+\alpha^{i}\mathbf{I}\right)^{-1}(\widetilde{\mathbf{S}}^{i})^{\mathrm{T}}. \tag{35}$$

Computationally, it is better not to compute the Kalman gain matrix, since it has a big dimension $N_x \times N_d$. However, for using localization based on tapering of the Kalman gain, the Kalman gain matrix needs to be computed.

Localization is implemented as a tapering of the values contained in the Kalman Gain matrix. For instance, one can define a tapering function that equals one at the measurement location and zero far from the measurement location where there should be no significant impact of the measurement. A common way of implementing localization is to compute the update from the following equation

$$\mathbf{X}^{i+1}=\mathbf{X}^{i}+\mathbf{C}\circ\mathbf{K}^{i}\widetilde{\mathbf{D}}^{i}. \tag{36}$$

Here the Schur or Hadamard product $\mathbf{C}\circ\mathbf{K}^i$ defines an element-wise multiplication and results in a new matrix of the same dimension as the original Kalman Gain matrix $\mathbf{K}^i \in \Re^{N_x \times N_d}$. Therefore, it is possible to use different linear combinations to update the uncertain model variables (Chen and Oliver, 2017).

One alternative to deal with the big dimension of the Kalman gain is to use Eq. (36) and sparsely represent the observed data, so it is possible to achieve a much lower number of observed data (N_d) and, then, one can deal with the Kalman gain matrix.

Another alternative is to define the effective Kalman gain matrix ($\mathbf{K}^i_{\mathrm{eff}}$) (Lorentzen et al., 2019; Luo and Bhakta, 2020; Chen and Oliver, 2017). For that purpose, if one considers Eq. (32) and compute the exact Singular Value Decomposition of $\widetilde{\mathbf{S}}^i$ as in

$$\widetilde{\mathbf{S}}^{i}=\mathbf{U}^{i}\mathbf{\Sigma}^{i}(\mathbf{V}^{i})^{\mathrm{T}}, \tag{37}$$

one will obtain the following equation

$$\mathbf{X}^{i+1}=\mathbf{X}^{i}+\mathbf{A}^{i}\mathbf{V}^{i}\left((\mathbf{\Sigma}^{i})^{\mathrm{T}}\mathbf{\Sigma}^{i}+\alpha^{i}\mathbf{I}\right)^{-1}(\mathbf{\Sigma}^{i})^{\mathrm{T}}(\mathbf{U}^{i})^{\mathrm{T}}\widetilde{\mathbf{D}}^{i}. \tag{38}$$

Therefore, one can define $\mathbf{K}^i_{\mathrm{eff}}$ as

$$\mathbf{K}^{i}_{\mathrm{eff}}=\mathbf{A}^{i}\mathbf{V}^{i}\left((\mathbf{\Sigma}^{i})^{\mathrm{T}}\mathbf{\Sigma}+\alpha^{i}\mathbf{I}\right)^{-1} \tag{39}$$

and apply localization

$$\mathbf{X}^{i+1}=\mathbf{X}^{i}+\mathbf{C}^{i}_{\mathrm{eff}}\circ\mathbf{K}^{i}_{\mathrm{eff}}(\mathbf{U}^{i}\mathbf{\Sigma}^{i})^{\mathrm{T}}\widetilde{\mathbf{D}}^{i}, \tag{40}$$

where $\mathbf{C}_{\mathrm{eff}}$ has dimension of $N_x \times N_e$ (same as the $\mathbf{K}_{\mathrm{eff}}$). The advantage of using this formulation is that by first multiplying the last three matrices $\left((\mathbf{U}^{i}\mathbf{\Sigma}^{i})^{\mathrm{T}}\widetilde{\mathbf{D}}^{i}\right)$, the effective Kalman gain matrix $\left(\mathbf{K}^{i}_{\mathrm{eff}}\right)$ will have dimension of $N_x \times N_e$, which is numerically more efficient to store and manipulate as $N_e \ll N_d$. Furthermore, as mentioned previously, this multiplication can be viewed as a projection of the observation residuals into a smaller subspace. However, there is some information loss, as we will show later (cf. Fig. 9).

There are different ways of computing $\mathbf{C}$. One of the most common approaches appears to be distance-dependent localization (Hamill et al., 2001; Emerick, 2016; Sakov and Bertino, 2011), in which all data points and model variables are assumed to be associated with certain physical locations. Besides, one needs to define a critical length to specify the regions in which each data should influence the model variables.

Nevertheless, Luo and Bhakta (2020) pointed out that the definition of the critical length might be difficult and case-dependent. Moreover, some types of observations used in data assimilation, like seismic data represented in a different transform domain, may not have any associated physical locations, in contrast to the more conventional production data from wells. To overcome these (and a few other) noticed issues, Luo and Bhakta (2020) proposed an automatic and adaptive correlation-based localization scheme.

3.3.1. Automatic and adaptive correlation-based localization

Luo and Bhakta (2020) calculated the correlation (ρ) between uncertain model variables in the initial ensemble ($\mathbf{X}^0$) and the initial simulated data ($\mathbf{g}(\mathbf{X}^0)$). Then, they defined threshold values to determine which observed data should be used to update which uncertain model variables. Since in practical implementations of ensemble-based methods, the ensemble size is limited (usually around 100), the estimated correlation from a limited sample between two uncorrelated variables might be different from zero. Therefore, the main idea in Luo and Bhakta (2020) is to obtain the spurious correlations (or noise) of these uncorrelated data, and use them to calculate the threshold values which are needed for the computation of $\mathbf{C}$.

To estimate the noise level, one can calculate the correlation fields between an ensemble of reservoir models (independent and identically distributed) and an ensemble of simulated data that are produced independently of the previous ensemble. Due to the independence between them, one can take the resulting correlation fields as the desired noise fields ε, which can be approximated by some Gaussian distributions with zero mean and unknown standard deviations denoted by σ hereafter. To calculate σ, Luo and Bhakta (2020) proposed to use the median absolute deviation estimator (Donoho and Johnstone, 1995), as in

$$\sigma = \frac{\text{median}(\text{abs}(\varepsilon))}{0.6745}. \tag{41}$$

After that, one can compute the threshold (θ) by

$$\theta = \sigma\sqrt{2\ln(\#\rho)}, \tag{42}$$

where $\#\rho$ is the number of elements in ρ.

Note that one should perform this procedure for each group type of uncertain variables. In other words, one must calculate the noise, standard deviation, and threshold for each group, such as porosity, horizontal permeability, and vertical permeability, for instance.

Finally, to generate a smooth tapering function, Luo and Bhakta (2020) defined a dummy variable

$$z = \frac{1 - \text{abs}(\rho)}{1 - \theta}, \tag{43}$$

where ρ is the sample correlation between a model variable and a data point, and θ is the corresponding threshold value obtained from Eq. (42). The variable z is then used in the Gaspari and Cohn formula (Gaspari and Cohn, 1999)

$$c(z) = \begin{cases} -\frac{1}{4}z^5 + \frac{1}{2}z^3 + \frac{5}{8}z^3 - \frac{5}{3}z^2 + 1, & \text{if } 0 \le z \le 1 \\ -\frac{1}{12}z^5 - \frac{1}{2}z^4 + \frac{5}{8}z^3 + \frac{5}{3}z^2 - 5z + 4 - \frac{2}{3}z^{-1}, & \text{if } 1 < z \le 2 \\ 0, & \text{if } z > 2. \end{cases} \tag{44}$$

Here, c are the entry values of the localization matrix $\mathbf{C}$.

Note that if one uses Eq. (36), one should calculate the correlation between the initial uncertain model variables ($\mathbf{X}^0$) and the simulated data $(\mathbf{g}(\mathbf{X}^0))$. However if one uses Eq. (40), the correlation should be computed between the initial uncertain model variables and the projected data $\left((\mathbf{U}^i\mathbf{\Sigma}^i)^{\mathrm{T}}(\mathbf{g}(\mathbf{X}^0))\right)$. For that reason, it is necessary to compute $\mathbf{C}^i_{\text{eff}}$ at each iteration, according to $(\mathbf{U}^i\mathbf{\Sigma}^i)^{\mathrm{T}}$.

It is important to highlight that when we use Eq. (36) and the sparsely observed data, we might still achieve a large N_{d} and, consequently, $\mathbf{K}^i$ might still be difficult to store and manipulate. Hence, one can compute each row or group of rows separately, or apply local analysis to deal better with the size of $\mathbf{K}^i$ (Sakov and Bertino, 2011; Chen and Oliver, 2017; Evensen et al., 2019; Brusdal et al., 2003). Nevertheless, we do not conduct local analysis in this work, but we will consider it in the future.

4. Observed data — sparse representation: Dictionary learning through the K-SVD algorithm

Nomenclature	
Δx	Step size in the x-direction
Δy	Step size in the y-direction
γ	Column vector of the Sparse matrix — sparse vector
$\mathbf{\Gamma}$	Sparse matrix
ϵ	Error tolerance
λ	Penalty for image reconstruction
σ	Noise standard deviation
c	Inner product coefficient between the atoms of the Dictionary and the residual
C	Constant in the calculation of the error tolerance
$\mathbf{d}$	Atom in the dictionary
$\mathbf{D}$	Dictionary
$\mathbf{E}$	Residual matrix
$\mathbf{g}$	Row vector of the Sparse matrix
$\mathbf{l}$	Data
L	Information loss
max	Maximum value
n	Patch size
N_{it}	Total number of iterations
N_{k}	Total number of atoms in the Dictionary
N_{nz}	Total number of nonzero coefficients
N_{ts}	Total number of patches (training dataset size)
$\mathbf{r}$	Residual vector
$\mathbf{x}$	Patches in the original seismic dataset
$\mathbf{X}$	Seismic dataset
Subscripts	
F	Frobenius
i	Patch index
I	Indexes of the locations of nonzero coefficients in $\mathbf{g}^{\mathrm{T}}$
j	Dictionary index
J	Collection of indexes of the atoms in the dictionaries used to calculate the sparse vector
Superscripts	
+	Pseudo-inverse
m	Iteration index for the sparse vector calculation
rec	Reconstructed
true	True
T	Transpose

As mentioned in the previous section, one alternative to deal with the big data from 4D seismic in history matching is through the application of sparse representation methods. Therefore, this part focuses on the Dictionary Learning method through the K-SVD algorithm (Aharon et al., 2006), which is responsible for sparsely represent the 4D seismic data to be used in history matching.

In the sparse representation problem, let us denote by $\mathbf{X}$ a matrix that is derived from the original seismic dataset (more information on how to form $\mathbf{X}$ is given in the next paragraphs). Besides, we assume that we have an initial guess of a matrix $\mathbf{D}$ (called dictionary) whose columns are filled with a set of predefined redundant basis elements (called atoms), and a separate matrix $\mathbf{\Gamma}$ associated with $\mathbf{D}$. Hence, our purpose here is to approximate $\mathbf{X}$ as

$$\mathbf{X} \approx \mathbf{D}\mathbf{\Gamma}. \tag{45}$$

Given the dictionary $\mathbf{D}$, the matrix $\mathbf{X}$ is (approximately) represented by the non-zero elements of the matrix $\mathbf{\Gamma}$. Therefore, if $\mathbf{\Gamma}$ is a sparse matrix, then one can achieve the purpose of sparse representation.

For numerical efficiency in handling big datasets, Aharon et al. (2006) suggested that a big dataset be first divided into a number of (possibly overlapping) subsets (called patches), so one can form $\mathbf{X}$. Note that in general, each patch can be a multidimensional (e.g., 2D or 3D) array extracted from the original seismic data (represented by the yellow box named "Observed data" on the right-hand side of Fig. 1). Here, we first consider 2D arrays each with dimension $n \times n$ where n denotes the patch size. We then reshape these 2D arrays into column vectors $\mathbf{x}_i \in \mathbb{R}^{n^2}$ for $i = 1, 2, \ldots, N_{\text{ts}}$, such that

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_{\text{ts}}}] \in \mathbb{R}^{n^2 \times N_{\text{ts}}}, \tag{46}$$

where N_{ts} is the number of patches (also called the training dataset size).

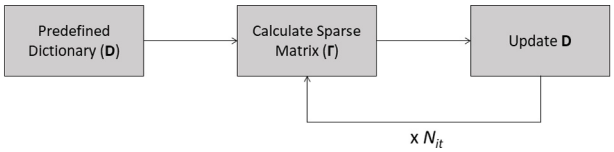

Fig. 2. Dictionary learning scheme.

Accordingly, let $\mathbf{D} \in \mathbb{R}^{n^2 \times N_k}$, where N_k denotes the number of atoms (columns) in the dictionary (called dictionary size), then one has the associated matrix $\mathbf{\Gamma}$ defined as

$$\mathbf{\Gamma} = [\gamma_1, \gamma_2, \dots, \gamma_{N_{ts}}] \in \mathbb{R}^{N_k \times N_{ts}}, \tag{47}$$

where γ_i $(i = 1, 2, \dots, N_{ts})$ are columns vectors that contain the representation coefficients associated with individual atoms. In other words, each $\mathbf{x}_i$ are approximated by a linear combination of the atoms in $\mathbf{D}$, in terms of

$$\mathbf{x}_i \approx \mathbf{D}\gamma_i. \tag{48}$$

In the current work, we adopt the K-SVD algorithm (Aharon et al., 2006) to tackle the sparse representation problem. Aharon et al. (2006) originally proposed K-SVD as a Dictionary Learning method for image denoising. Later on, this algorithm has also been applied to different areas, for instance, face recognition problems (Zhang and Li, 2010), magnetic resonance image reconstruction (Ravishankar and Bresler, 2011), and representation of geological facies in oil reservoirs (Liu and Jafarpour, 2013).

K-SVD is an iterative algorithm and can be divided into two main parts: (1) calculation of the sparse representation matrix; and (2) dictionary update. As such, one needs to pre-choose a dictionary to initialize the algorithm. Elad and Aharon (2006) and Soares et al. (2019) used a Discrete Cosine Transform (DCT) function as the initial dictionary and reported it as a good choice. Note that the initial dictionary can also be in other forms, such as wavelet or curvelet basis functions, or some random parts of the original image. Through the learning process, the dictionary will be updated to better adapt to each specific problem on hand, and consequently, to achieve a better representation with fewer non-zero coefficients (Turquais, 2018). Fig. 2 depicts the scheme for the Dictionary Learning method, and as an iterative method, one should define the total number of iterations N_{it}.

4.1. Calculation of the sparse representation matrix

After the initial dictionary is chosen, one can move to calculate the sparse matrix $\mathbf{\Gamma}$. As suggested by Aharon et al. (2006), it is necessary to divide the original dataset into smaller patches and transform them into column vectors $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_{ts}})$. Subsequently, one can calculate the sparse vector for each patch $(\gamma_1, \gamma_2, \dots, \gamma_{N_{ts}})$.

One way of solving the sparse representation problem is to add an error constraint (Rubinstein et al., 2008) as in

$$\gamma_i = \underset{\gamma_i}{\text{Argmin}} \ \|\gamma_i\|_0 \quad \text{subject to} \quad \|\mathbf{x}_i - \mathbf{D}\gamma_i\|_2^2 \le \epsilon\,. \tag{49}$$

Here ϵ specifies the error tolerance, and is calculated as

$$\epsilon = \left(C\sigma\sqrt{n^2}\right)^2, \tag{50}$$

where C is a constant determined as 1.15 by Elad and Aharon (2006), and σ is the noise standard deviation, which needs to be defined beforehand.

The approach proposed by Elad and Aharon (2006) is to use the Orthogonal Matching Pursuit (OMP) to iteratively fill the sparse vector γ_i with one non-zero coefficient each time, until the constraint $(\|\mathbf{x}_i - \mathbf{D}\gamma_i\|_2^2 \le \epsilon)$ is satisfied.

In the OMP algorithm, the first step is to calculate the inner product coefficient between the columns (or atoms) $\mathbf{d}_j$ $(j = 1, 2, \dots, N_k)$ of the dictionary $\mathbf{D}$ and the residual $\mathbf{r}_i^m$ $(i = 1, 2, \dots, N_{ts})$

$$c_i^m = (\mathbf{r}_i^m)^{\mathrm{T}}\mathbf{d}_j, \tag{51}$$

where m is the iteration index for the sparse vector calculation, and the residual $\mathbf{r}_i^m$ is defined as

$$\mathbf{r}_i^m = \mathbf{x}_i - \mathbf{D}\gamma_i^m. \tag{52}$$

Note that for the first iteration $\gamma_i^0 = 0$, then $\mathbf{r}_i^m = \mathbf{x}_i$.

Next, one selects the atom with the highest inner product coefficient with $\mathbf{r}_i^m$ and stores its index in J. In other words, one selects the atom with the largest correlation with the residual. Finally, it is necessary to orthogonalize the original data $(\mathbf{x}_i)$ in the space spanned by the atoms in J (denote by $\mathbf{D}_J$), in terms of

$$\gamma_i^m = (\mathbf{D}_J)^{+}\mathbf{x}_i = (\mathbf{D}_J^{\mathrm{T}}\mathbf{D}_J)^{-1}\mathbf{D}_J^{\mathrm{T}}\mathbf{x}_i. \tag{53}$$

One should do these steps multiple times until reaching the stopping criterion. It is important to highlight that at every iteration step (m), one adds one index to J and one non-zero coefficient to γ_i, and these iterations are related only to the calculation of the sparse matrix.

4.2. Dictionary update

After calculating the sparse matrix $\mathbf{\Gamma}$, one needs to update the dictionary $\mathbf{D}$. In this regard, Elad and Aharon (2006) proposed to do the update column by column. Hence, one should only consider the signals in $\mathbf{X}$ that uses the atom being updated. In addition, Elad and Aharon (2006) also proposed to update only the non-zero coefficients in $\mathbf{\Gamma}$.

If one thinks of $\mathbf{\Gamma}$ as a matrix composed of the row vectors $\mathbf{g}_j^T$ and considers updating the jth atom $\mathbf{d}_j$ of the Dictionary $\mathbf{D}$, one can calculate the residual

$$\mathbf{E}_j = \mathbf{X}_{I_j} - \sum_{a \ne j} \mathbf{d}_a \mathbf{g}_{a,I_j}^T, \tag{54}$$

when the $\mathbf{d}_j$ atom is excluded for sparse representation. The list I_j indexes the locations of non-zero coefficients in $\mathbf{g}_j^T$. For instance, if

$$\mathbf{g}_j^T = (0, \gamma_{j,2}, 0, \dots, 0, \gamma_{j,10}, 0, \dots, 0, \gamma_{j,22}, 0, \dots, 0),$$

then $\mathbf{g}_{j,I_j}^T = (\gamma_{j,2}, \gamma_{j,10}, \gamma_{j,22})$. Therefore, one can expect that

$$\mathbf{E}_j = \mathbf{X}_{I_j} - \sum_{a \ne j} \mathbf{d}_a \mathbf{g}_{a,I_j}^T \approx \mathbf{d}_j \mathbf{g}_{j,I_j}^T. \tag{55}$$

Consequently, one can update $\mathbf{d}_j$ and $\mathbf{g}_{j,I_j}^T$ by solving the following minimization problem

$$\text{Armin}_{\mathbf{d}_j, \mathbf{g}_{j,I_j}^T} \left\|\mathbf{E}_j - \mathbf{d}_j \mathbf{g}_{j,I_j}^T\right\|_F^2 \quad \text{subject to} \quad \left\|\mathbf{d}_j\right\|_2 = 1\,, \tag{56}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

This problem can be solved by Singular Value Decomposition (SVD) or by an approximate solution proposed by Rubinstein et al. (2008), where they accelerate the optimization process by seeking a sub-optimal solution instead through the following equations

$$\mathbf{d}_j = \mathbf{X}_{I_j}\mathbf{g}_{j,I_j} - \left(\sum_{a \ne j} \mathbf{d}_a \mathbf{g}_{a,I_j}^T\right)\mathbf{g}_{j,I_j}\,. \tag{57}$$

$$\mathbf{d}_j = \mathbf{d}_j / \|\mathbf{d}_j\|_2\,. \tag{58}$$

$$\mathbf{g}_{j,I_j}^{\mathrm{T}} = \mathbf{d}_j^{\mathrm{T}}\mathbf{X}_{I_j} - \mathbf{d}_j^{\mathrm{T}}\left(\sum_{a \ne j} \mathbf{d}_a \mathbf{g}_{a,I_j}^T\right). \tag{59}$$

More information about the method can be found in Rubinstein et al. (2008) and Soares et al. (2019).

To check if the sparse representation matrix is able to capture the main characteristics of the data, one can calculate the reconstructed dataset by

$$\mathbf{X}^{\text{rec}} = \mathbf{D}\mathbf{\Gamma}, \tag{60}$$

and compare if $\mathbf{X}^{\text{rec}}$ and $\mathbf{X}$ are similar. Luo et al. (2018) used the following measure of information loss (L)

$$L = \frac{\|\mathbf{I}^{\text{true}} - \mathbf{I}^{\text{rec}}\|_2}{\|\mathbf{I}^{\text{true}}\|_2} \times 100, \tag{61}$$

where $\mathbf{I}^{\text{true}}$ is the quantity containing the true or reference data (after excluding noise, if any) and $\mathbf{I}^{\text{rec}}$ the quantity with respect to the reconstructed data.

If one uses the 4D AVA dataset in seismic history matching, most of the data may just stem from 4D noise. Hence, the non-zero coefficients of the sparse matrix $\mathbf{\Gamma}$ may be concentrated on some specific parts of the matrix. Consequently, the reconstructed image may contain certain discontinuities. In order to reconstruct a smoother image, instead of using Eq. (60), Elad and Aharon (2006) proposed the following formula

$$\mathbf{X}^{\text{rec}} = (\lambda \mathbf{I})^{-1}(\lambda \mathbf{X} + \mathbf{D}\mathbf{\Gamma}), \tag{62}$$

which represents a weighted average between the reconstructed dataset ($\mathbf{D}\mathbf{\Gamma}$) and the original one. In Eq. (62), λ is a penalty (or weight) parameter, which in this work is chosen as

$$\lambda = \frac{\max(\mathbf{X})}{10\sigma}, \tag{63}$$

following the suggestion of Rubinstein et al. (2008), where max is the maximum value of a property.

4.3. Parameters in the K-SVD method

From the previous introduction, one can see that there are several parameters involved in the K-SVD method, such as dictionary size (N_{k}), training dataset size (N_{ts}), patch size (n), and number of iterations (N_{it}). Depending on the objective and the dataset in the case study, the optimal values of these parameters may differ.

Relevant to the current study, Soares et al. (2019) evaluated how each of these parameters affects the number of non-zero coefficients retained in the sparse matrix (N_{nz}) and the quality of the reconstructed image using a 4D seismic dataset. According to the authors, the most influential parameter on the number of non-zero coefficients is the training dataset size. The lower N_{ts} is, the smaller N_{nz} becomes. Hence, to make N_{nz} as small as possible, it is better to collect patches in the dataset without any overlapping between them. If one uses a higher N_{ts}, the quality of the reconstructed image tends to be better. However, one will end up with a larger number of non-zero coefficients.

To demonstrate how to select the training dataset according to the patch size, Fig. 3 shows a case with regular patches in the size of 4×4 (denoted by 4 hereafter). In the figure, the green color corresponds to the case in which one generates patches without any overlapping in-between, and thus achieves the minimum training dataset size N_{ts}. In this case, the step size $\Delta x \times \Delta y$, which is determined by the distances of a moving patch that travels along horizontal (x) and vertical (y) directions, is equal to the patch size (e.g., 4×4 in the current example). It is important to highlight that one should travel along each axis at a time to be able to encompass the whole grid. Meanwhile, in the same figure, we also use the red color to indicate the case with the maximum training dataset size N_{ts}, where the step size corresponds to 1×1 (denoted by 1 hereafter). Note that for image reconstruction, one should average the values from the overlapped patches.

Another important parameter is the dictionary size N_{k}, i.e., the number of atoms in the dictionary. Soares et al. (2019) showed that the higher N_{k} is, the smaller N_{nz} one can achieve. However, the maximum number of N_{k} cannot be larger than the training dataset size N_{ts}.

In addition, there is a big trade-off concerning the number of iterations (N_{it}), since a larger N_{it} can indeed reduce the number N_{nz} of non-zero coefficients, but at the cost of an increased computational time. Finally, as discussed in Aharon et al. (2006), larger image patches do not tend to work well with the K-SVD algorithm, and Soares et al. (2019) reported that lower values of n can help to achieve lower N_{nz}.

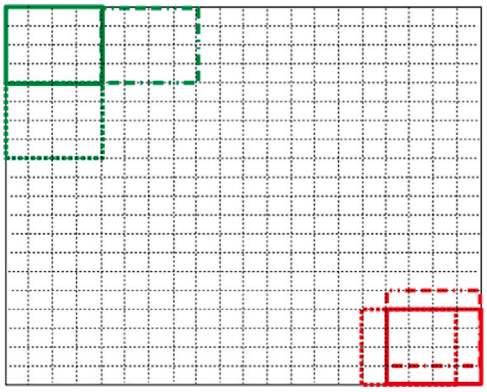

Fig. 3. Example of training dataset and image patches. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The end products of the K-SVD algorithm are the values of the non-zero coefficients, their locations in $\mathbf{\Gamma}$ and the learned dictionary $\mathbf{D}$, which will be utilized in the subsequent 4D seismic history matching problem.

4.4. Sparse representation — forward simulation

Sparse representation of the simulated data follows a similar procedure as that for the observed data. After obtaining sparse representation for the observed data, one should save the Dictionary $\mathbf{D}$ of the iteration $N_{\text{it}} - 1$ and the location of the non-zero coefficients in the final $\mathbf{\Gamma}$.[1] Hence, one can use the same procedure detailed in the sub Section 4.1 "Sparse Matrix Definition" and select the coefficients in the simulated $\mathbf{\Gamma}$ at the same position of the non-zero coefficients in the observed $\mathbf{\Gamma}$. Note that even if there are other non-zero coefficients in the simulated $\mathbf{\Gamma}$, they will not be considered.

5. Case study: Brugge field

The case study used in this work is the Brugge benchmark (Peters et al., 2010), which consists of a reservoir with four production zones, two high and two low permeable zones. The two high permeable zones are in layers 1–2 and 6–8, with the averaged permeability values being 810 mD and 1105 mD, respectively. The two low permeable zones are in layers 3–5 and 9, with the averaged permeability values being 90 and 36 mD, respectively. Fig. 4 shows the distributions of the permeability (along the x-direction) on each layer, with respect to one of the reservoir models from the initial ensemble. In addition, this benchmark is an isotropic case, i.e., the permeability in the y-direction is the same as in the x-direction. Note that we did not show the initial porosity maps for succinctness.

The numerical reservoir model of the Brugge Field has $139 \times 48 \times 9$ cells, summing up to 60 048 gridblocks in total, among which 44 550 are active. The model contains 20 producer and 10 injector wells, with water being the only injected fluid. Fig. 5 shows the distribution of the wells, where the injectors are located in the border and the producers are more centralized. The benchmark case has 10 years of production data, and we use a black oil simulator (ECLIPSE) for reservoir simulation.

There are 104 different realizations of reservoir models provided in the benchmark case. In this work, we randomly pick one of the models

[1] The reason for us to use the dictionary $\mathbf{D}$ at the iteration step $N_{\text{it}} - 1$, instead of N_{it}, is that in our code implementation, the sparse matrix $\mathbf{\Gamma}$ at the final iteration step N_{it} is obtained by solving the sparse representation problem Eq. (49), using $\mathbf{D}$ at the iteration step $N_{it} - 1$.

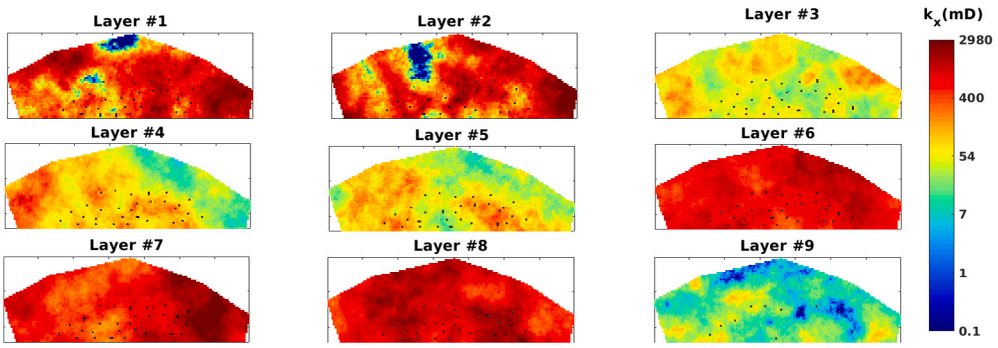

Fig. 4. Horizontal permeability (k_x) from realization number 1. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

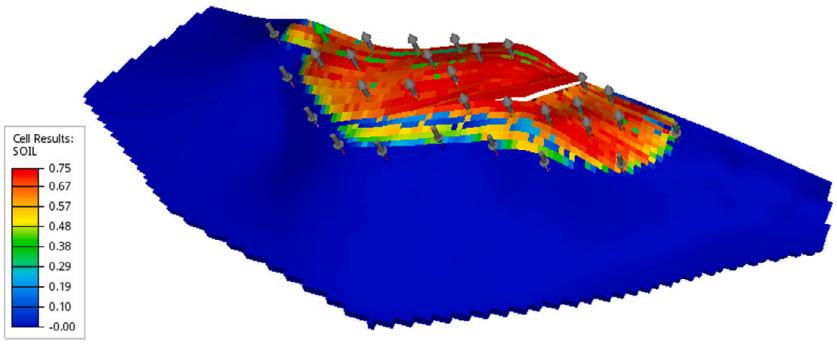

Fig. 5. Distribution of the wells. The grid indicates the initial oil saturation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

as the reference case to generate the "observed" seismic data, and let the rest compose the initial ensemble of 103 models. In each reservoir model from the initial ensemble, the uncertain parameters include the porosity and permeability in the x-, y-, and z-directions.

The seismic attributes in our experiments are the amplitude-versus-angle (AVA) data with two different offset angles, near (10°) and far (20°), from 3 different surveys: base (day 1), monitor #1 (day 991), and monitor #2 (day 2999) (Luo et al., 2018). The AVA dataset is on a seismic scale, so it has a different dimension compared to the reservoir model. In our experiments, the dimension of the AVA dataset at each offset angle and at each survey time is 139 × 48 × 176, summing up to a total of 1 174 272 data points. Therefore, the total number of data points in our 4D seismic dataset is 7 045 632 (6 × 1 174 272).

To generate the AVA data, we require over- and under-burden layers and properties in addition to the reservoir model. Furthermore, the seismic trace/signal is recorded in some specific sampling rate in the time domain (z-axis in Fig. 6), which makes the dimension of the seismic traces (in time-domain) larger. In this work, we consider the whole cube of the AVA data as observed data (Fig. 6), however, most of the parts of the cubes are coming from over-burden and under-burden areas (non-reservoir parts). Therefore, the AVA dataset has a larger dimension if we compare it to the simulation model. Note that in applications with acoustic impedance datasets, the data is upscaled to the same dimension of the simulation model or even in a lower dimension, in which one can use fewer seismic horizons that are correlated with more than one layer in the simulation model, for instance.

Table 1 summarizes the main characteristics of the Brugge benchmark case, together with some of the experimental settings in the current work.

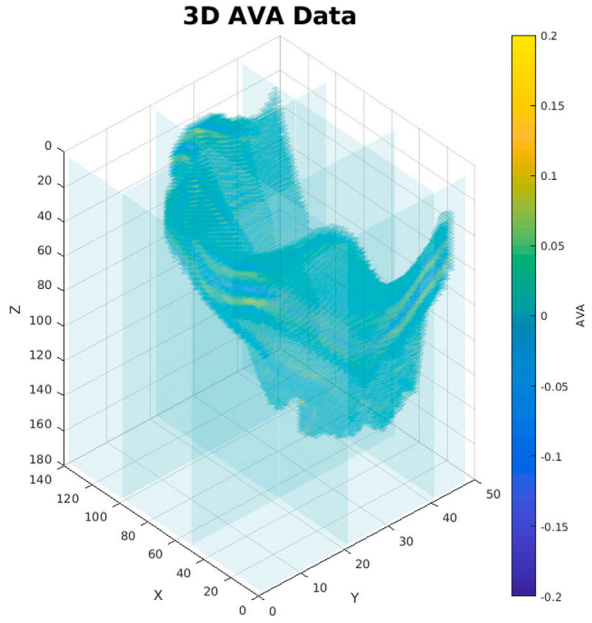

Fig. 6. 3D AVA data (observed data): far trace in the base survey (day 1). The color bar in the right indicates the values of AVA data points. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Methodology and application

In the current work, we directly use the seismic attributes at three surveys as the observations in data assimilation. In principle, one may also try using the differences between different surveys as the observations. This would help to reduce the data size of 4D seismic, on the other hand, though, the signal-to-noise ratios in these difference datasets may become much lower, as is observed in our experiments (cf. upper left plot in Fig. 22).

Table 1
Information summary about the Brugge benchmark case study.

Model dimension	$139 \times 48 \times 9$ (60 048 cells)
Active cells	44 550
Average gridblock size	93 m × 91 m × 5 m
Wells	20 producers and 10 injectors
Production data	10 years (3 647.5 days)
4D seismic data	Near (10°) and far (20°) offsets amplitude-versus-angle (AVA) data
Seismic survey time	Base (day 1), Monitor #1 (day 991), Monitor #2 (day 2 999)
Seismic dimension	7 045 632 coefficients ($6 \times 139 \times 48 \times 176$)
Model variables	Porosity + Permeability along the x-,y- and z-directions. Total number: 178 200

As indicated in Fig. 7, we divide our investigations into two main parts. In the first part, we evaluate the necessity of using sparse representation in a 4D seismic data assimilation problem by comparing the assimilation performance resulting from using the Projected Seismic Dataset (PSD) and a Sparse Seismic Dataset (SSD) obtained from using the K-SVD algorithm. While in the second one, we aim to improve the data assimilation performance by using the K-SVD algorithm differently, where we first use the K-SVD algorithm to suppress the noise in the 4D seismic dataset and then use the same algorithm again for sparse data representation. For distinction, we call the end product of this procedure (using K-SVD twice) Denoised Sparse Seismic Dataset (DSSD).

For history matching, we use the IES-RML with the formulation shown by the Eq. (32). Note that with this formulation, we do not need to compute the truncated singular value decomposition, as most of the previous works did. In addition, we use the aforementioned three stopping criteria, where the maximum number of iterations is set to 20. Finally, we start with $\beta^0 = 1$, with the reduction and increment factor being 0.9 and 2, respectively.

Through the experiments in the three cases (PSD, SSD, and DSSD), we first evaluate if sparse representation can improve the performance of data assimilation at the expense of increased computational time and information loss. Additionally, we also inspect if denoising 4D seismic datasets before sparse representation can help to further improve the performance of data assimilation.

In the following, we explain in more detail how we perform data assimilation with respect to each approach (PSD, SSD, and DSSD).

6.1. Projected seismic dataset (PSD)

In this case, we use the full seismic dataset, i.e., the near and far offset of the three surveys, and project them onto the subspace, in the form of $\left(\mathbf{U}^i\mathbf{\Sigma}^i\right)^{\mathrm{T}}\widetilde{\mathbf{D}}^i$, as Eq. (40) suggests, for data-size reduction. In this way, we can use the whole dataset without incurring any computer-memory-related issue in the course of updating uncertain parameters, since we need to store the $\mathbf{K}_{\mathrm{eff}}$, which has a dimension of only $N_{\mathrm{x}} \times N_{\mathrm{e}}$. Note that in this case, we adopt the automatic and adaptive correlation-based scheme (Luo and Bhakta, 2020) for localization, in which the tapering matrix is constructed based on the correlations between model variables and projected data.

To estimate the measurement errors involved in data assimilation, we use the k-means clustering method (Soares et al., 2019), in which the seismic data are grouped into 3 clusters, one representing the noise and the other two containing positive and negative values that are more likely to be informative signals than noise. We can then calculate the standard deviation of the noise cluster, and use it in Eq. (50) and to construct the observation error covariance matrix ($\mathbf{C}_{\mathrm{d}}$) for data assimilation. Note that Obidegwu et al. (2017) and Davolio and Schiozer (2018) also used k-means, but to cluster acoustic impedance data in softening and hardening effects, and Coleou et al. (2012) pointed out other works that used k-means to cluster seismic waveforms for facies classification. Since the noise added consists of white noise, we assume $\mathbf{C}_{\mathrm{d}}$ to be diagonal, i.e., there is no correlation between the observation error. However, if one faces colored noises, it is possible to use the wavelet formulation to determine the noise as in Lorentzen et al. (2020). We chose here to use k-means because of its simplicity and good results reported previously.

Table 2
K-SVD parameters in SSD.

Parameter	Value
Patch size (n)	8
Step size	8
Iterations (N_{it})	20
Dictionary size (N_{k})	750
Dictionary type	DCT

Therefore, by projecting the whole 4D seismic dataset onto a subspace, it is possible to avoid the use of sparse representation while also achieving a significant data-size reduction. Note that in this case, the whole 4D seismic data, including the noise itself that does not contain information of the reservoir conditions, will be assimilated into reservoir models. This may downgrade the assimilation performance, as will be shown later.

6.2. Sparse seismic dataset (SSD)

To sparsely represent the 4D seismic dataset, we use the K-SVD method introduced before to find a representation with as few coefficients as possible while preserving the main characteristics of the data. In the experiments, we adapted the Matlab toolbox developed by Rubinstein et al. (2008) to use the K-SVD algorithm. The original toolbox can be found at http://www.cs.technion.ac.il/~ronrubin/software.html. Based on the analysis of Soares et al. (2019), we select the parameters involved in the calculation of the sparse matrix, as shown in Table 2. Note that the patch size and step size are equal in all three dimensions, i.e., $8 \times 8 \times 8$.

As we only use the non-zero coefficients during the data assimilation and, consequently, compress the data, it is possible to handle the Kalman gain without projecting them into the subspace of $\left(\mathbf{U}^i\mathbf{\Sigma}^i\right)^{\mathrm{T}}$ as in the previous approach. Therefore, we use the formulation in Eq. (36) in this step.

For the calculation of the $\mathbf{C}_{\mathrm{d}}$, we use the same procedure as before. Note that this is an approximation since we find the error in the original observed data and not in the sparsely represented data. The readers are referred to Raanes et al. (2019) for more information about the error statistics and problems with correlated errors.

6.3. Denoised sparse seismic dataset (DSSD)

Elad and Aharon (2006) originally developed the K-SVD as a denoising tool. In line with this initiative, we apply the K-SVD algorithm in two steps: the first one (denoising) involves using the algorithm to achieve the best-reconstructed image without considering the number of non-zero coefficients N_{nz} retained. In the second step (compression), we apply the algorithm one more time but now focus on getting the best-reconstructed image with as few coefficients as possible. By suppressing noise in the original dataset, we expect to have seismic datasets with better quality, which in turn may help achieve improved assimilation performance.

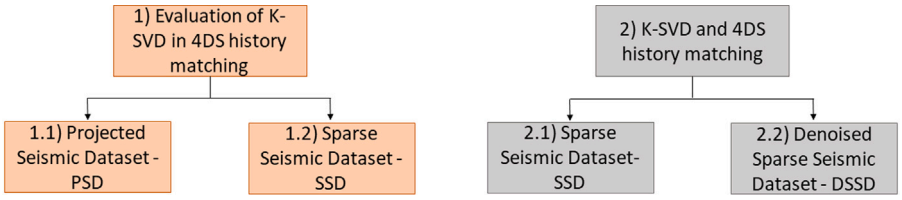

Fig. 7. The two main parts in our investigations.

Table 3
K-SVD configuration in the DSSD case.

Parameter	Denoising	Compression
Patch size (n)	8	8
Dimension	2D	3D
Step size	1	8
Iterations (N_{it})	10	50
Dictionary size (N_k)	256	750
Dictionary type	Data	Data

Table 4
Seismic noise.

Seismic data	Calculated σ	True σ	Noise level (%)
Far trace — base	0.0077	0.0074	32.92
Near trace — base	0.0154	0.0146	33.25
Far trace — monitor 1	0.0076	0.0073	32.89
Near trace — monitor 1	0.0156	0.0149	33.14
Far trace — monitor 2	0.0076	0.0072	32.85
Near trace — monitor 2	0.0156	0.0148	33.08

Due to the different objectives, the way we apply the K-SVD algorithm differs. Table 3 summarizes the configurations of the K-SVD algorithm in these two steps.

For the purpose of denoising, we treat each set of seismic attributes as a collection of 2D images and sequentially apply the K-SVD algorithm to denoise one 2D image at each time. Since the patch dimension for a 2D image is smaller, we believe that handling a 2D image each time helps to accelerate the process, as shown by Soares et al. (2019). Furthermore, as in this step we are not concerned about the number of non-zero coefficients, using a smaller patch size would not affect the efficiency of sparse representation. On the other hand, for the purpose of compression, we directly apply the K-SVD algorithm to 3D seismic datasets.

In the denoising step, we choose to use the maximum training dataset size N_{ts}, as this helps to generate better-reconstructed images. In contrast, in the compression step, we use the minimum N_{ts} instead to obtain fewer non-zero coefficients. For the same reason, the number of iterations is higher in the compression step, as this also helps to reduce the number of non-zero coefficients. In addition, we choose to increase the dictionary size to obtain less N_{nz}.

Instead of using DCT as the initial dictionary, we select random patches from the training dataset to construct our initial dictionary. Note that DCT is a more general approach, while the random patches is more specific for each case, which can help to generate fewer non-zero coefficients in the sparse matrix, as we will show later.

We use the same formulation for the history matching as before, Eq. (36), and to define the $\mathbf{C}_d$, we use the denoised version of the observed data.

7. Results and discussions

7.1. Results comparison between the PSD and SSD cases

The upper plot in Fig. 8 shows a sample inline seismic attribute, e.g., a vertical cross-section in the y–z plane from the far offset 3D AVA data at the base survey (day 1) (Fig. 6) and with the index number of x being 80. The first step is to define the noise in the seismic dataset and to estimate it, we use the k-means algorithm to cluster the seismic data into three regions (noise, positive, and negative values), as the lower plot in Fig. 8 illustrates. Then, we calculate the standard deviation (σ) of Cluster 1, which represents the noise. Table 4 shows the values of σ for each seismic data and the associated noise levels, which we calculate as in

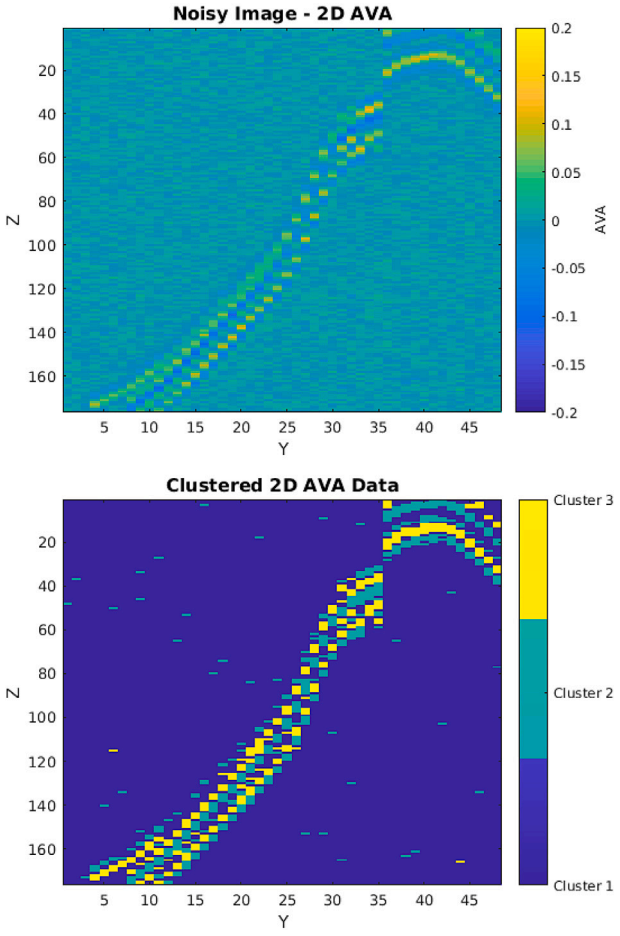

Fig. 8. Far trace AVA data — base survey: Noisy image (upper); Clustered image (lower). The horizontal axis represents the y-dimension as in Fig. 6 and the vertical axis represents the z-dimension as in Fig. 6. The color bar in the right indicates the AVA values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\text{Noise level} = \frac{\text{noise variance}}{\text{pure signal variance}} \times 100\%. \tag{64}$$

Luo et al. (2018) stated that the noise level in the reference case is 30%. Hence, we observe that with k-means clustering, we achieved noise levels around 33%, which are close to that in the reference case.

In the PSD case, we select the full seismic dataset and project it into the smaller subspace $\left(\mathbf{U}^i\mathbf{\Sigma}^i\right)^{\mathrm{T}}$ as mentioned previously. Therefore, if we

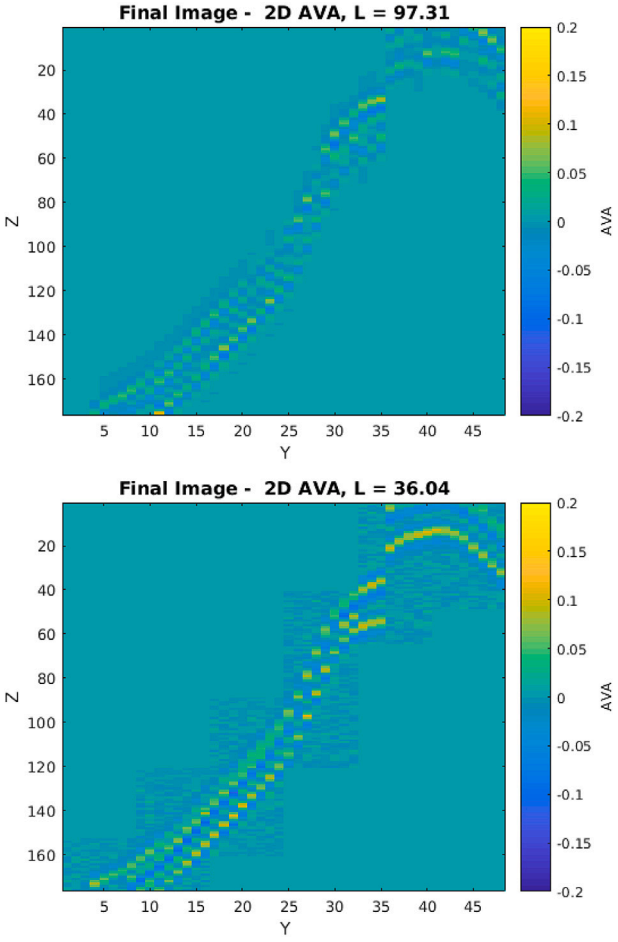

Fig. 9. Reconstructed far trace AVA data — base survey: PSD (upper); SSD (lower). The horizontal axis represents the y-dimension as in Fig. 6 and the vertical axis represents the z-dimension as in Fig. 6. The color bar in the right indicates the AVA values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

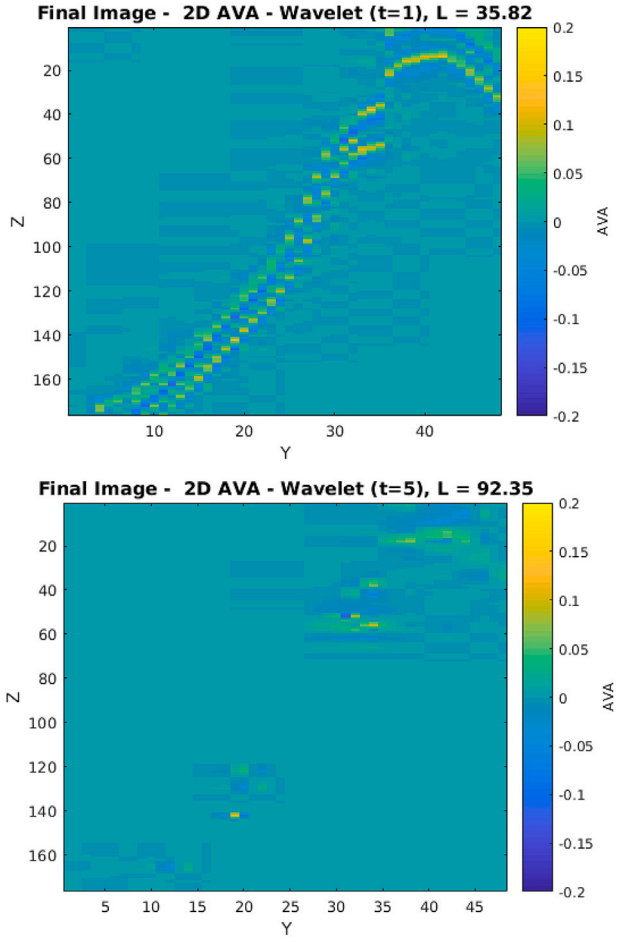

Fig. 10. Reconstructed far trace AVA data — base survey — Wavelet based: $t = 1$ (upper); $t = 5$ (lower). The horizontal axis represents the y-dimension as in Fig. 6 and the vertical axis represents the z-dimension as in Fig. 6. The color bar in the right indicates the AVA values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

consider a random observed data ($\mathbf{d}_j$) and project it onto $\left(\mathbf{U}^0\mathbf{\Sigma}^0\right)^{\mathrm{T}}$, we would achieve the following projected data

$$\mathbf{d}_j^{\mathrm{proj}} = \left(\mathbf{U}^0\mathbf{\Sigma}^0\right)^{\mathrm{T}}\mathbf{d}_j. \tag{65}$$

Hence, it is possible to reconstruct the signal ($\mathbf{d}_j^{\mathrm{rec}}$) and check how much information we loose as in

$$\mathbf{d}_j^{\mathrm{rec}} = \left(\left(\mathbf{U}^0\mathbf{\Sigma}^0\right)^{\mathrm{T}}\right)^{+}\mathbf{d}_j^{\mathrm{proj}}. \tag{66}$$

The upper plot in Fig. 9 depicts the reconstructed image after the projection and the information loss L is calculated through Eq. (61).

For the SSD approach, we use the characteristics described in Table 2 and calculate the error (Eq. (50)) using σ defined by the k-means. The lower plot in Fig. 9 shows the reconstructed image after applying the K-SVD algorithm in the SSD approach, where we calculate $\mathbf{X}^{\mathrm{rec}}$ through Eq. (60). Note that we use Eq. (60) once we want to show only the characteristics retained by the method.

If we compare the two plots in Fig. 9 with the original AVA dataset (Fig. 8), we notice that they both preserved the structure of the signal. However, we achieve a better-reconstructed image for the SSD case. The explanation for that lies in the fact that the projection can be interpreted as a smooth function. It is possible to see this if one looks at the region with values very close to zero in the original image (upper plot in Fig. 8), and this region has values of zero in the reconstructed image (upper plot in Fig. 9). Hence, this smooth property will also happen in the whole dataset, including the part that we have values that are not too close to zero. In addition, the ensemble may not be able to accommodate all the information contained in the observed data, and the iterative learning process in the SSD helped to generate a better-reconstructed image. Consequently, the projection downgraded the image reconstruction compared to the SSD case.

After the application of the K-SVD algorithm for the entire dataset (far and near offset traces at base, first monitor and second monitor), we achieved a total of 19 055 coefficients, which represents only 0.27% of the entire dataset.

Even though the scope of this work is not to compare different methodologies for sparse representation, we show some differences between the wavelet-based sparse representation (Luo et al., 2018) and the K-SVD based one. In the methodology developed by Luo et al. (2018), there is a need for defining a threshold value (t) that is used to select the non-zero coefficients for sparse representation. Fig. 10 depicts the case for $t = 1$ (upper plot) and for $t = 5$ (lower plot). The focus here is on the trade-off between the value of t and the quality of the reconstructed image. When t equals 5, the image does not retain the main features of the original seismic data. However, the number of non-zero coefficients retained was only 3 293 (0.047% of the original seismic dataset). For the case where t is equal to 1, the final reconstructed image is very similar to the original data, but the amount of non-zero coefficients is much larger, up to 178 332 (2.53% of the original dataset).

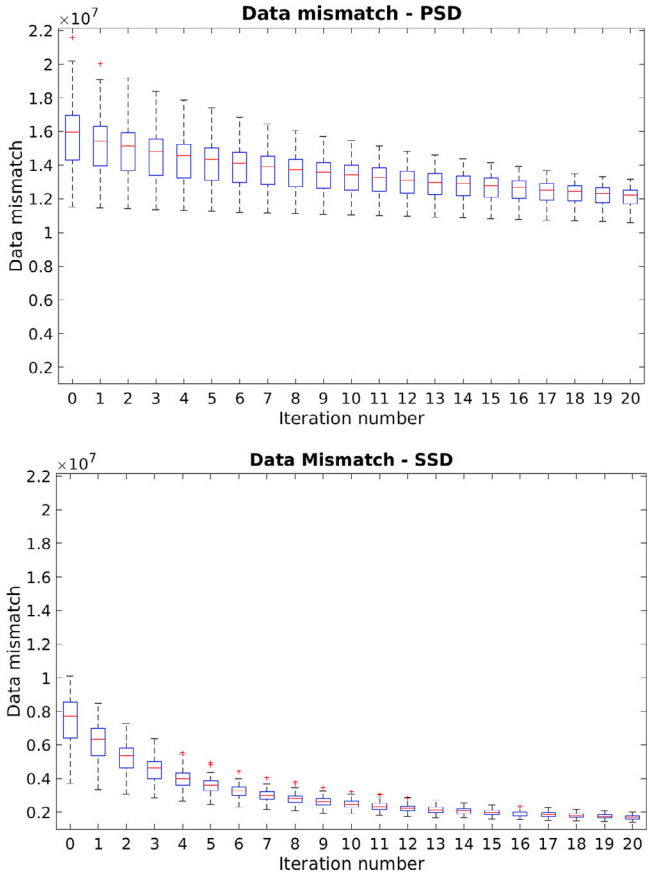

Fig. 11. Data mismatch: Projected Seismic Dataset — PSD (upper); and Sparse Seismic Dataset — SSD (lower).

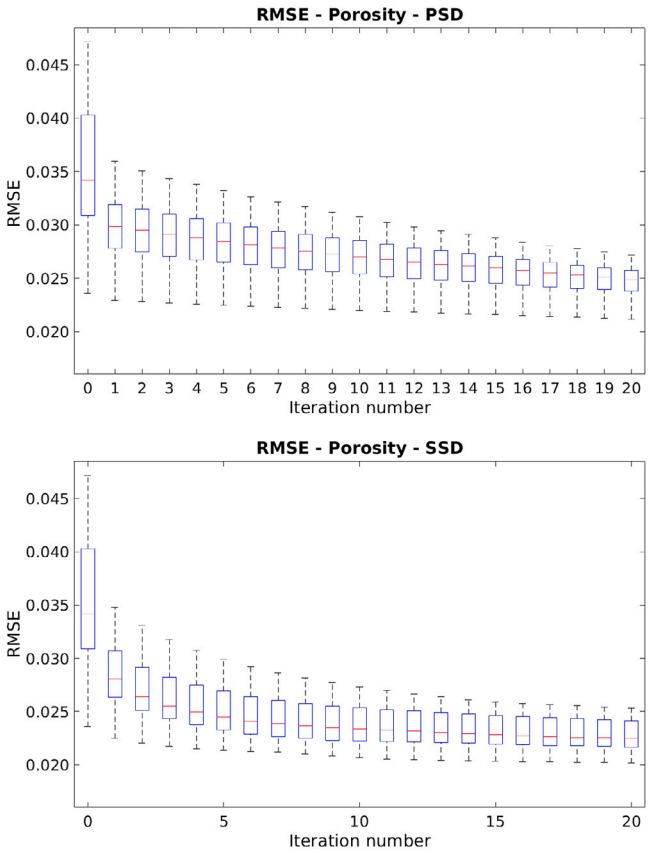

Fig. 12. RMSE — Porosity: Projected Seismic Dataset — PSD (upper); and Sparse Seismic Dataset — SSD (lower).

Therefore, in this case, the K-SVD algorithm tends to generate better-reconstructed images with fewer non-zero coefficients, while for the wavelet-based sparse representation it remains to be a challenge to find a suitable trade-off between the retained characteristics in the reconstructed data and the number of non-zero coefficients.

The analysis of data assimilation performance starts with the calculation of the data mismatch through Eq. (23) without the sum, i.e., we calculate the data mismatch for each ensemble member and plot in a boxplot for every iteration as Fig. 11 depicts. We observe that both approaches tend to reduce the mismatch between the simulated and observed data. In the PSD case (Fig. 11 - upper plot), it is possible to see that the rate of mismatch reduction tends to be slower than that in the SSD case (Fig. 11 - lower plot). Nevertheless, it is difficult to directly compare the data mismatch from both cases as we use different sets of observed data. Note that we could normalize the mismatch, but in the PSD case, most of the observed data concerns the non-reservoir part. Hence, during the forward simulation, we set these values to zero, and consequently, as the observed data in these parts are also very close to zero, the normalized mismatch tends to be much lower for the PSD case. Therefore, it is better to show how the estimated model variables look like in the final ensemble.

We use the Root Mean Square Error (RMSE), as in Luo et al. (2018), to measure the distances between the estimated model variables and the reference (true) model.

Fig. 12 shows the RMSE of porosity values for PSD (upper) and SSD (lower). In both cases, RMSE values tend to decrease as the iterations proceed. One can see that using SSD leads to lower RMSE values at the end, i.e., models closer to the reference one.

By inspecting the ensemble mean for the PSD and SSD cases (third and fourth plots in Fig. 13, respectively), one can see that there is a stronger update of the porosity values in the latter case. The ensemble mean obtained in the PSD case remains closer to the mean of the initial ensemble (first plot in Fig. 13), while the one from the SSD case is closer to the reference model (second plot in Fig. 13).

More insights can be gained by inspecting the tapering matrix produced by the localization scheme. Note that the tapering matrix is in the same dimension as the Kalman Gain matrix. Hence, in the PSD case, it is a matrix in the dimension of $N_x \times N_e$. In contrast, in the SSD case, it has the dimension of $N_x \times N_d$, where N_d is the number of observations (in this case, the number of non-zero coefficients). Following this perspective, each column of a tapering matrix indicates how an observation element is correlated with the model variables. Since in our experimental settings, the PSD and SSD cases have different amounts of observations, we check the mean of columns and plot the tapering coefficient values $c(z)$ (Eq. (44)) onto the reservoir model gridblocks.

Fig. 14 shows the tapering coefficient values using correlations between porosity and the selected observation element, distributed on the reservoir model gridblocks at Layer 2, in cases of PSD (upper) and SSD (lower). In the PSD case (Fig. 14 - upper), the tapering coefficient values are relatively low. In contrast, in the SSD case (Fig. 14 - lower), the tapering coefficient values tend to be higher. Therefore, the final ensemble of the PSD case tends to remain closer to the initial one, while the one from the SSD case experiences stronger updates and moves closer to the reference model.

We also examine RMSEs concerning the permeability. Fig. 15 shows that the RMSE of permeability (along the x-direction) does not change as much as porosity since seismic data tend to be less sensitive to permeability in this benchmark case (Luo et al., 2018). Similar results are observed for the estimated permeability values along the y- and z-directions. We do not show the permeability maps for succinctness.

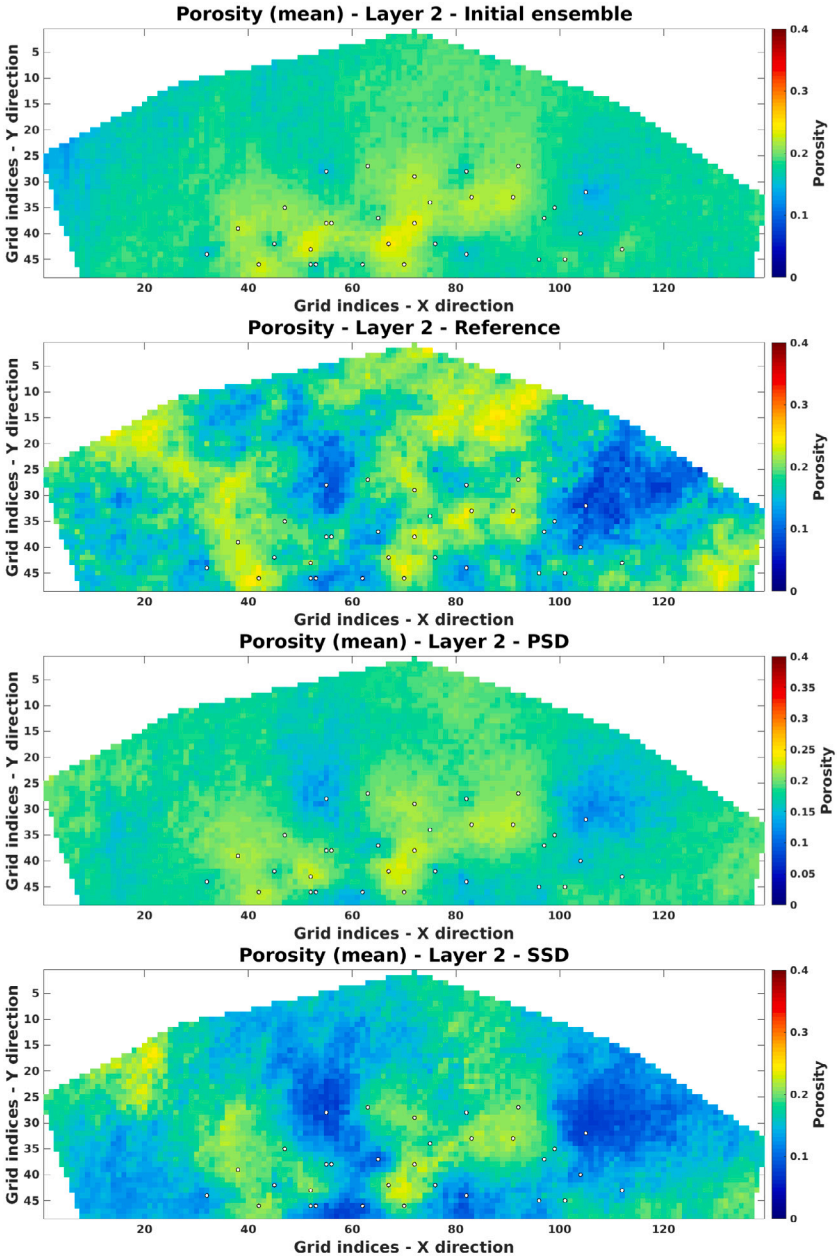

Fig. 13. Porosity maps at Layer 2 of the reservoir. From top to bottom: initial ensemble mean; reference map; final ensemble mean from the PSD case; final ensemble mean from the SSD case. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In conclusion, projecting observations onto an ensemble subspace and sparse data representation (through K-SVD) may serve as two viable ways to handle big data assimilation problems in practice. In this benchmark case study, it shows that sparse data representation tends to result in better assimilation performance than the subspace-projection based method.

7.2. Results comparison between the SSD and DSSD cases

The results from the SSD case are the same as in the preceding subsection. We choose to show some of the results again here for comparison.

In the DSSD case, we first denoise the original dataset using the K-SVD algorithm, whose configuration is indicated in the middle column of Table 3. We use the same estimated noise standard deviations σ as before and apply the K-SVD algorithm to 2D images (slices in the x-direction) 139 times. The upper plot in Fig. 16 illustrates the image obtained by applying the K-SVD algorithm for the purpose of denoising. One can see that information loss ($L = 13.61$) is smaller than that in the SSD case ($L = 36.04$). Note that we use Eq. (62) to promote continuity in the reconstructed image.

Denoising of seismic data is a very important topic within the geoscience community. Consequently, there are many works focused on this particular matter. For instance, Baddari et al. (2011) developed a non-linear diffusion filter capable of reducing random and Gaussian noise, Xiong et al. (2014) proposed a random noise attenuation in the time–frequency domain, and Hennenfent and Herrmann (2006) presented the non-uniformly sampled curvelets to denoised seismic images. There is a vast list in the literature with different methodologies, and since it is not the scope of this work to compare different denoising methods, we can refer the readers more works about this topic (Latif and Mousa, 2017; Shan et al., 2009; Han and van der Baan, 2015; Bonar and Sacchi, 2012; Zhu et al., 2019).

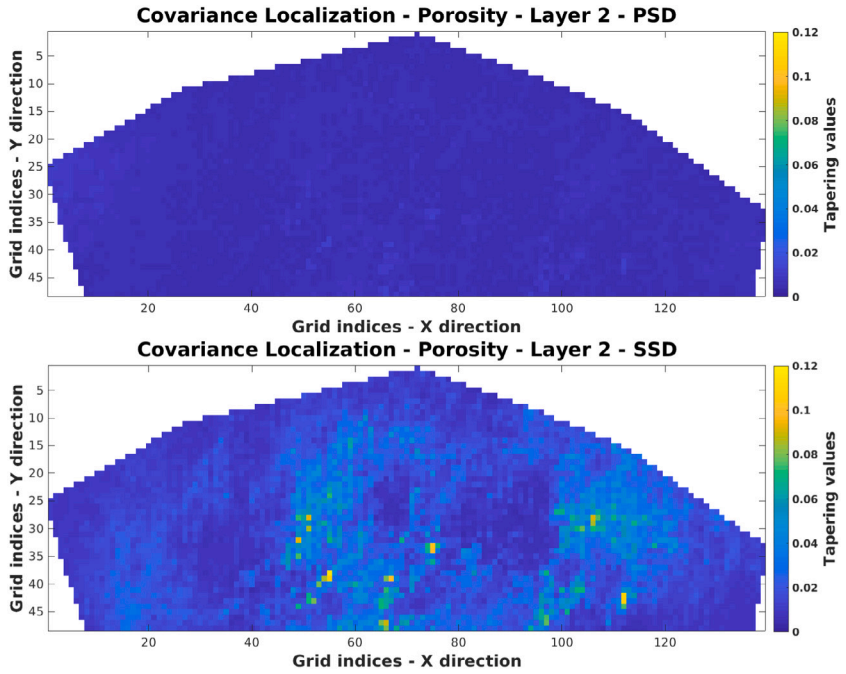

Fig. 14. Tapering coefficient values (for porosity) distributed at Layer 2 of the reservoir model gridblocks: Projected Seismic Dataset — PSD (upper); and Sparse Seismic Dataset — SSD (lower). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

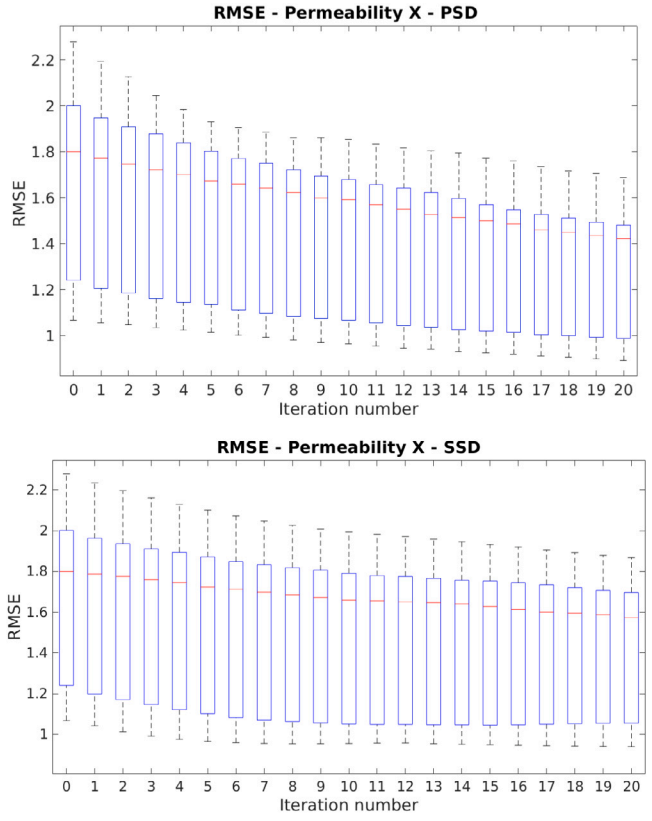

Fig. 15. RMSE — Permeability x: Projected Seismic Dataset — PSD (upper); and Sparse Seismic Dataset — SSD (lower).

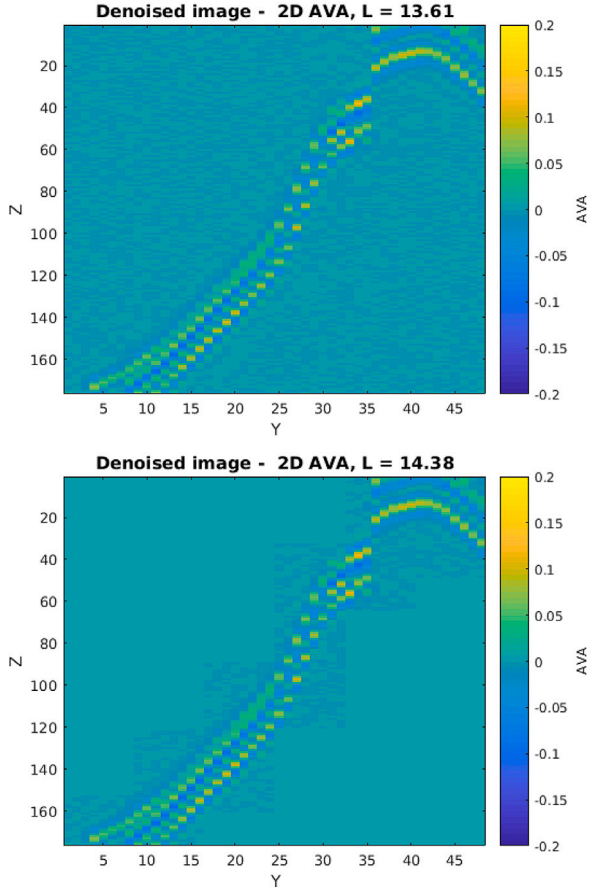

Fig. 16. Far trace AVA data - base survey: Denoised data (upper); and Reconstructed data from a sparse representation of the denoised data (lower). The horizontal axis represents the y-dimension as in Fig. 6 and the vertical axis represents the z-dimension as in Fig. 6. The color bar in the right indicates the AVA values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Subsequently, we use the k-means clustering method one more time to obtain new values of estimated noise standard deviations σ, which are reported in Table 5. Comparing the results there with those in

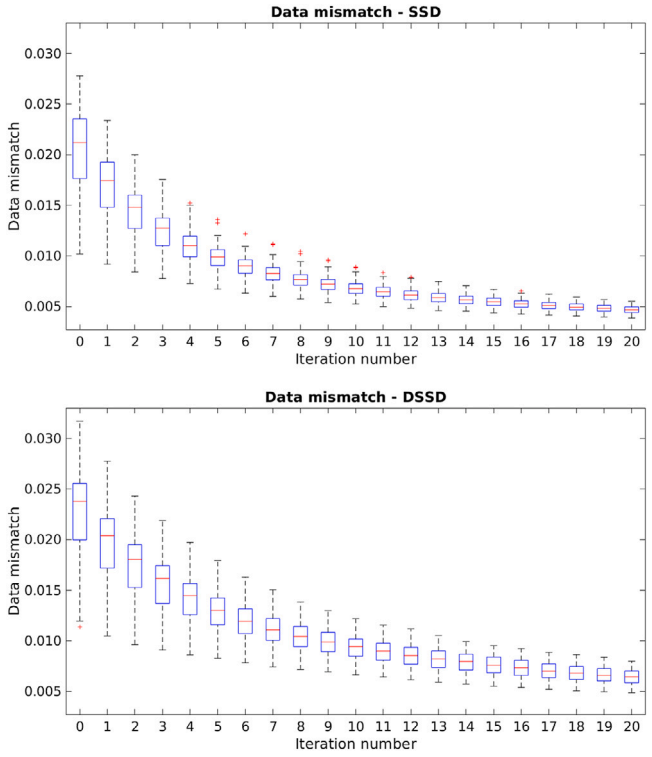

Fig. 17. Normalized data mismatch: Sparse Seismic Dataset — SSD (upper); and Denoised Sparse Seismic Dataset — DSSD (lower).

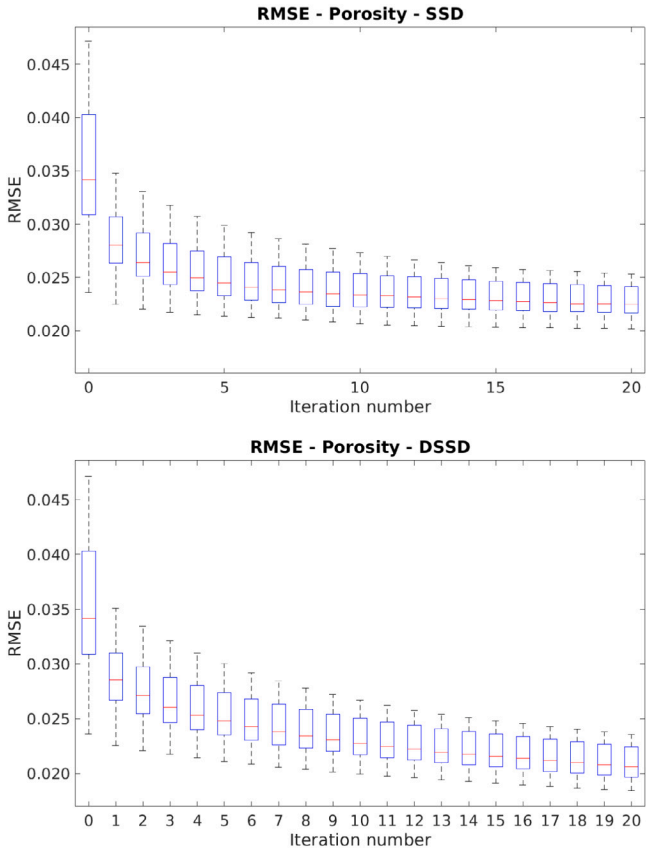

Fig. 18. RMSE — Porosity: Sparse Seismic Dataset — SSD (upper); and Denoised Sparse Seismic Dataset — DSSD (lower).

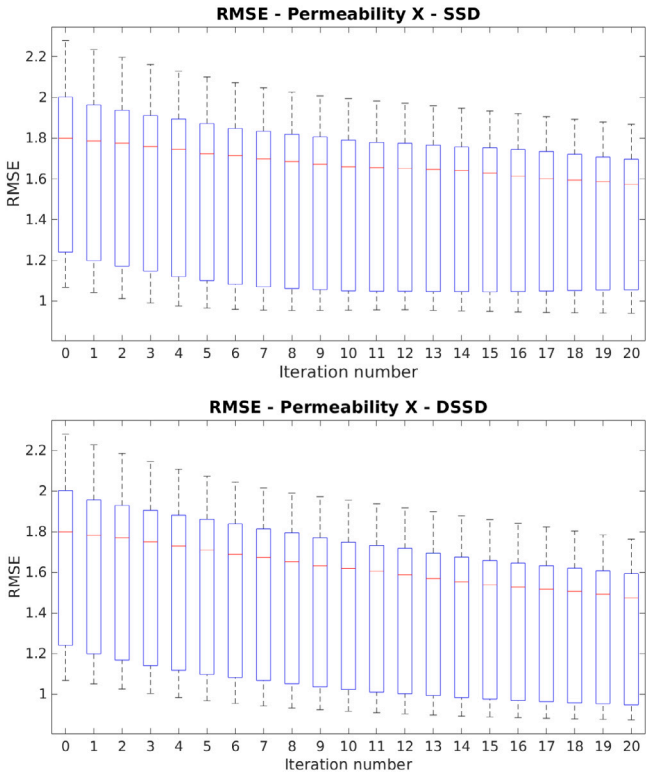

Fig. 19. RMSE — Permeability X: Sparse Seismic Dataset — SSD (upper); and Denoised Sparse Seismic Dataset — DSSD (lower).

Table 5
Seismic noise after denoising.

Seismic data	Calculated σ
Far trace — base	0.0034
Near trace — base	0.0070
Far trace — monitor 1	0.0033
Near trace — monitor 1	0.0071
Far trace — monitor 2	0.0033
Near trace — monitor 2	0.0070

Table 4, one can see that the new σ values are lower, indicating the impact from the preceding denoising procedure.

Note that we do not use DCT as the initial dictionary in the DSSD case. This is because during the second step (compression), we achieve a relatively large number of non-zero coefficients by using DCT (due to the lower error). However, when we use random patches from our original dataset to construct the initial dictionary (called data dictionary hereafter), we can reduce the number of non-zero coefficients. Therefore, to maintain the consistency between the denoising and compression procedures, we prefer to use the data dictionary in both steps. The information loss (L) of the reconstructed image from a sparse representation of the denoised data is slightly higher than that of the denoised data itself (cf. lower plot in Fig. 16). In this sense, we achieve minimal information loss within our experiment settings in the DSSD case.

After the denoising and compression procedures, we obtain a total amount of 38 876 non-zero coefficients, which represents 0.55% of the original dataset. Note that here we end up with about twice the number of non-zero coefficients in the SSD case. However, in terms of information loss, we obtain better-reconstructed images from sparse representations of the original dataset in the DSSD case (cf. the lower plot in Fig. 16).

The upper and lower plots of Fig. 17 illustrate the box plots of data mismatch at different iterations steps in the SSD and DSSD cases,

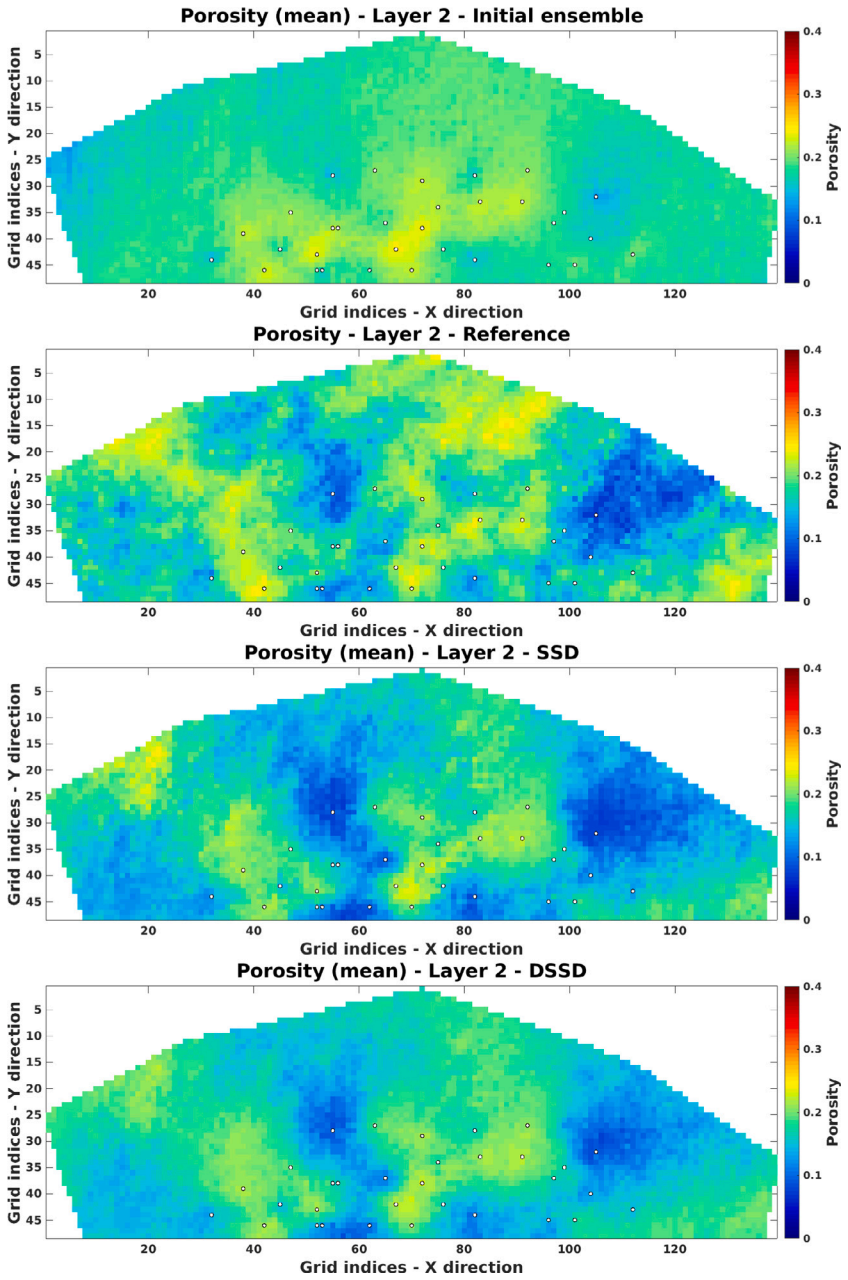

Fig. 20. Porosity maps at Layer 2 of the reservoir. From top to bottom: initial ensemble mean; reference map; final ensemble mean from the SSD case; final ensemble mean from the DSSD case. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

respectively. Since both cases now have observed data concerning only the reservoir part, it is possible to normalize the mismatch and directly compare them. As one can notice, SSD presented lower values for the mismatch. However, the difference between the two cases is not very big. Furthermore, even though the uncertainty in the mismatch is lower in the SSD, this does not reflect in the uncertain model variables (porosity and permeabilities) as the spread (standard deviation) of the final ensemble is very similar for both cases. To complement the analysis, we also compare the estimated reservoir models with the reference one.

Fig. 18 shows the boxplots of RMSE (for porosity) at different iterations steps in the SSD (upper) and DSSD (lower) cases. As one can see, the final RMSE values tend to be close in both cases, with those from the DSSD case tending to be lower. In addition, one can observe that in the SSD case, the RMSE values seem to enter a plateau from the 10th iteration step on, while (although not verified) those in the DSSD case appear to have room for further reduction, following the exhibited trend therein. For RMSEs with respect to the permeabilities, we obtain similar results to those in the SSD case, possibly due to the weak correlations between seismic data and permeabilities, as mentioned before (Fig. 19).

The third and fourth panels of Fig. 20 show the mean porosity maps (of the final ensembles) at Layer 2 of the reservoir in the SSD and DSSD cases, respectively. One can see that both porosity maps capture some of the prominent geological structures in the reference one (second panel of the same figure). Visually, these two estimated porosity maps are similar, although the RMSE metric (cf. Fig. 18) indicates that the one from the DSSD case tends to be better. Similar results are also found in the permeability (along the x-direction) maps, as demonstrated in Fig. 21.

After showing the relative superiority (in terms of RMSE) in using DSSD, we also consider some of the differences between the SSD and

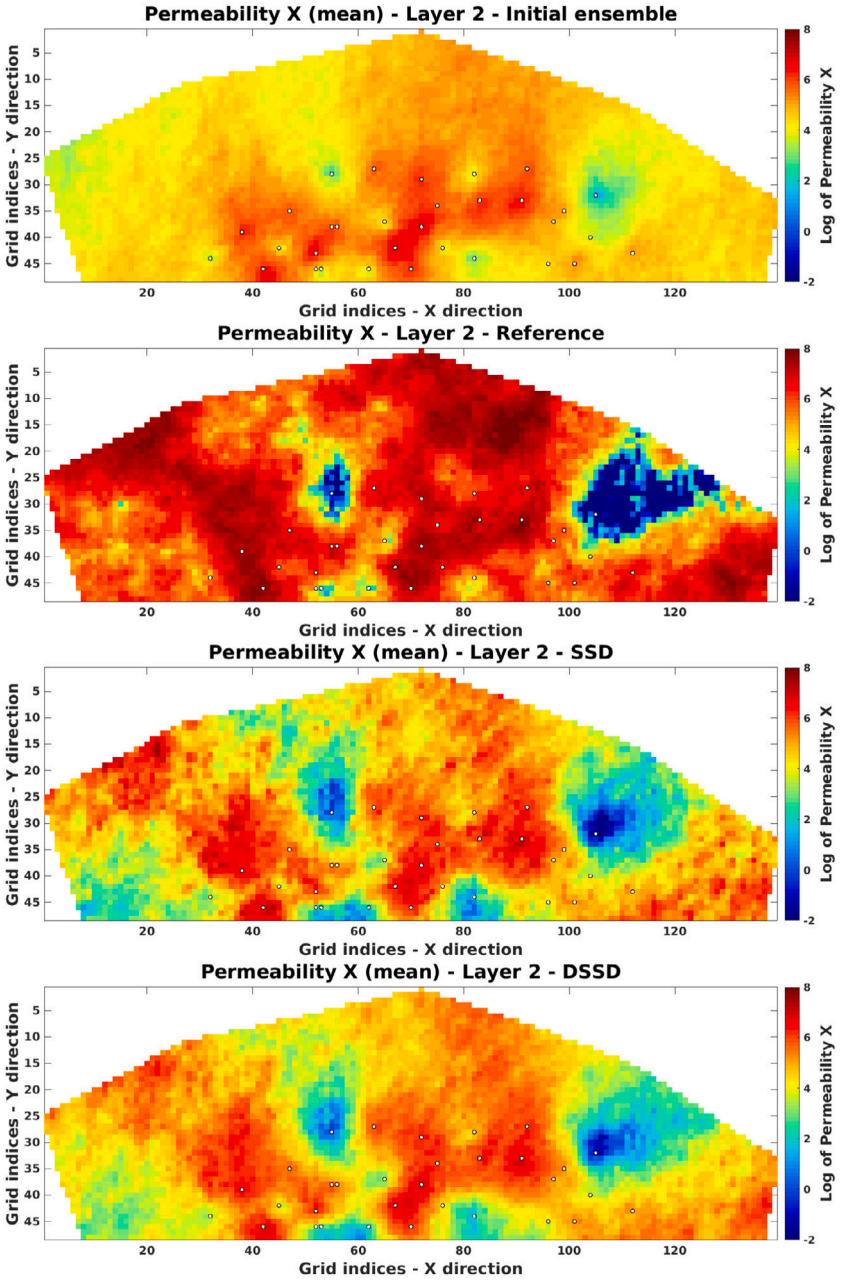

Fig. 21. Log of permeability x maps at Layer 2 of the reservoir. From top to bottom: initial ensemble mean; reference map; final ensemble mean from the SSD case; final ensemble mean from the DSSD case. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

DSSD cases from the perspective of computational resources. In the Brugge benchmark case, from the model side, one reservoir model contains 44 550 active gridblocks, and each gridblock has 4 uncertain model variables (porosity and permeabilities along three directions) to estimate. Hence, we have a total of 178 200 uncertain model variables (N_x) in data assimilation. In addition, from the observation side, we have 38 876 data points (N_d) in the DSSD case, resulting in a Kalman gain matrix (**K**) in the dimension of $178\,200 \times 38\,876$. In contrast, in the SSD case, the Kalman gain matrix (**K**) is in the dimension of $178\,200 \times 19\,055$ instead, which is somewhat easier to handle in terms of computational time and memory.

Regarding the computational time dedicated to sparse data representation, in the DSSD case, it takes about 40 min to go through both procedures of denoising and sparse representation. On the other hand, it takes only around 50 s in the SSD case. This substantial gap between the computational time is largely due to the time-consuming denoising procedure adopted in the DSSD case, where a minimum step size (1×1) is adopted so that it creates a much larger number of overlapping patches to achieve high-quality denoised images. Furthermore, one would need a relatively large RAM memory to handle all the matrices during history matching. For instance, to calculate the Kalman gain **K** for the SSD and DSSD cases, one would need about 25 and 50 GB, respectively. This number seems to be very big, but it can be avoided by computing the Kalman gain several times considering different lines of the matrix at each time (Emerick, 2016), and one can also use local analysis to avoid these big matrices (Brusdal et al., 2003; Evensen, 2009a; Chen and Oliver, 2017; Sakov and Bertino, 2011).

Finally, to demonstrate the benefits of using DSSD in the Brugge benchmark, we examine the 4D changes of seismic data reconstructed from the original dataset and their sparse representations. Fig. 22 shows the differences between a slice of far offset trace at the monitor survey #1 and the same slice of trace data at the base survey. In other words,

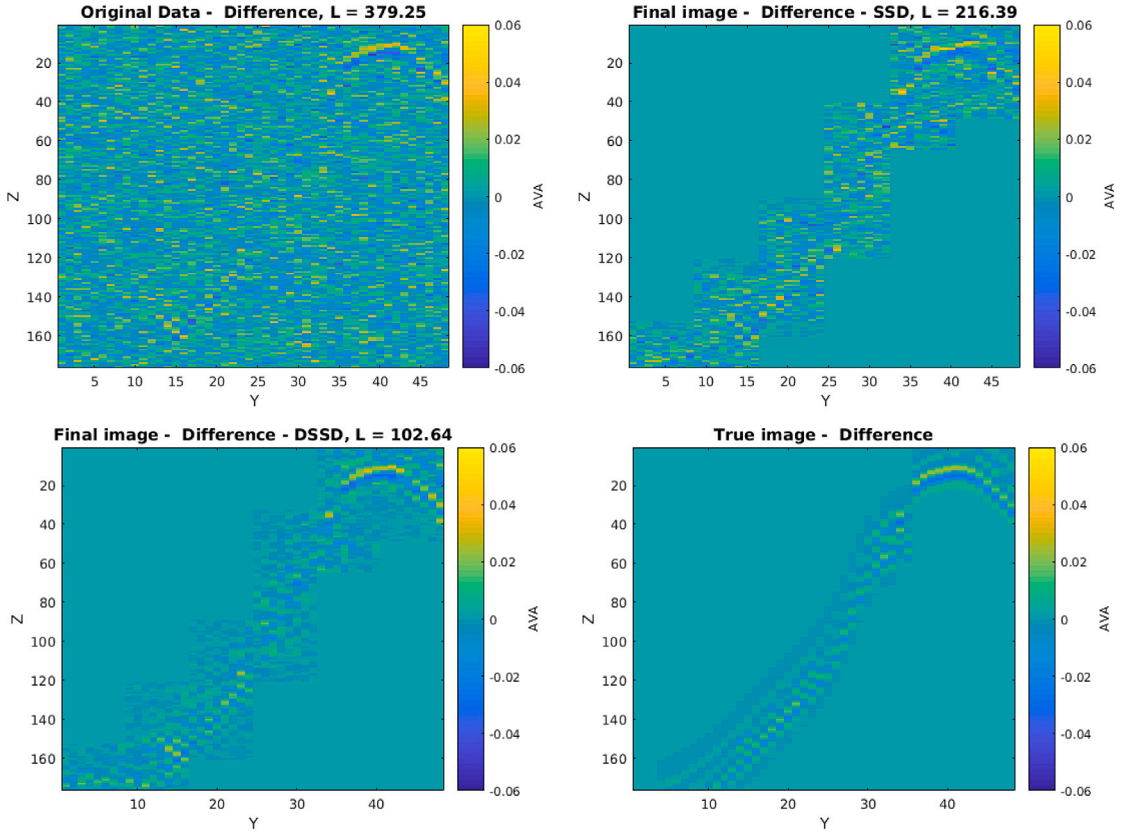

Fig. 22. Differences of a slice of far AVA trace data between monitor #1 and base surveys: original dataset (upper left); SSD (upper right); DSSD (lower left); reference case (lower right). The horizontal axis represents the y-dimension as in Fig. 6 and the vertical axis represents the z-dimension as in Fig. 6. The color bar in the right indicates the AVA values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

we subtract $\mathbf{X}^{\text{rec}}$ at monitor #1 from $\mathbf{X}^{\text{rec}}$ at the base survey in the SSD and DSSD cases. One can see that in the upper left panel, the difference between the two surveys in the original dataset is very noisy and very different from the reference case (lower right panel). In the SSD case (upper right panel), the difference is still big from the reference case, while the differences in the DSSD case (lower left panel) appear to be more similar.

When we inspect the 4D differences in the SSD case (upper right panel of Fig. 22) without the denoising procedure, the noise kept in the reconstructed data may sum up and downgrade the quality of the 4D seismic data. In turn, this may deteriorate the performance of data assimilation, if one wishes to use the 4D differences as the observations.[2]

In contrast, in the DSSD case (lower left panel of Fig. 22), by denoising the original seismic data first, we obtain reconstructed images with improved qualities (cf. Fig. 16). Consequently, the difference between the survey is more physically consistent, as we can see the 4D effect better, and the resulting 4D differences may resemble the reference case (lower right panel) better, which is also reflected by the lower RMSE values.

Therefore, despite higher computational time and memory required, adopting DSSD tends to capture the 4D effect better in the Brugge benchmark, resulting in a final ensemble closer to the reference case than using SSD. It is important to mention that for the Brugge benchmark, the original AVA data has a relatively low signal-to-noise ratio. Hence, the denoising process helped to achieve better-reconstructed images and better preservation of the 4D effect. However, some other cases may present a higher signal-to-noise ratio and a good visualization of the 4D effect without denoising the signal. In such cases, since the SSD case is faster, we suggest using SSD instead of DSSD. In other words, one can use different criteria to decide which case to use, one can either have a larger "weight" on the computational resources or on a better final model.

Besides, if we compare the results of the SSD and DSSD cases with that from the previous work (Luo et al., 2018; Luo and Bhakta, 2020), in which the authors used the wavelet-based sparse representation, the cases presented here tend to be more convenient in achieving a suitable trade-off between sparse data representation and preservation of data information.

8. Conclusions

We present a 4D seismic data assimilation framework in which a dictionary learning algorithm (K-SVD) is adopted for sparse data representation, instead of using a wavelet-based sparse representation procedure or deep learning as in previous works. Through numerical experiments in the Brugge benchmark case, we show that dictionary-learning based sparse representation can serve as an efficient way to handle big data assimilation problems.

We also consider an alternative way to handle big seismic data by projecting them onto an ensemble subspace. In the investigated benchmark case, it turns out that the projected data tend to exhibit relatively weak correlations with the model variables under estimation. As a result, within our experimental settings, the resulting final ensemble

[2] We note that an alternative way is to first calculate the 4D differences, apply the K-SVD algorithm, and then reconstruct the 4D differences from their sparse representations. However, as aforementioned, in this case, the signal-to-noise ratios in 4D difference data become very low such that it becomes very difficult to extract true seismic signals from the noisy data (upper left plot in Fig. 22).

of updated reservoir models remains close to the initial ensemble and thus leads to relatively inferior assimilation performance. In contrast, through dictionary-learning based sparse representation, the corresponding representation coefficients tend to have stronger correlations with the model variables under estimation. This makes it possible for the updated reservoir models to experience more substantial changes and move closer to the reference model.

In an additional investigation, we introduce a denoising procedure before applying dictionary-learning based sparse representation to the 4D seismic. Although incurring higher computational time and memory, our experiments in the Brugge benchmark indicate that this additional procedure helps to improve the performance of data assimilation in terms of the RMSE metric, in cases where the original seismic signal is noisy. However, if the original signal has a high signal-to-noise ratio (low noise), the denoising process might be less useful, since it incurs a higher computational time. Hence, in cases where the original signal is not very noisy and the computational resources are limited, the SSD case may be preferred.

Finally, even though we reduced the AVA dataset to only 0.55% of the original dataset in the DSSD case, this number still appears too big. Therefore, as our future work, we will consider applying local analysis (Evensen, 2009a; Sakov and Bertino, 2011; Chen and Oliver, 2017) to handle big datasets more efficiently. In addition, we also plan to test the approaches developed here in a different dataset for further performance validation.

CRediT authorship contribution statement

R.V. Soares: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Visualization. **X. Luo:** Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing - review & editing, Supervision. **G. Evensen:** Conceptualization, Methodology, Writing - review & editing, Supervision, Project administration, Funding acquisition. **T. Bhakta:** Methodology, Software, Data curation, Writing - review and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors acknowledge financial support from the Research Council of Norway through the Petromaks-2 project DIGIRES (RCN no. 280473) and the industrial partners AkerBP, Wintershall DEA, Vår Energi, Petrobras, Equinor, Lundin, and Neptune Energy. We also thank Schlumberger for providing academic software licenses to ECLIPSE.

References

Aanonsen, S.I., Nævdal, G., Oliver, D.S., Reynolds, A.C., Vallès, B., 2009. The ensemble Kalman filter in reservoir engineering–a Review. SPE J. 14 (3), 393–412. http://dx.doi.org/10.2118/117274-PA, URL: http://www.onepetro.org/doi/10.2118/117274-PA.

Aharon, M., Elad, M., Bruckstein, A., 2006. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans. Signal Process. 54 (11), 4311–4322. http://dx.doi.org/10.1109/TSP.2006.881199, arXiv:59749104367.

Alfonzo, M., Oliver, D.S., 2019. Seismic data assimilation with an imperfect model. Comput. Geosci. http://dx.doi.org/10.1007/s10596-019-09849-0.

Almeida, F.L.R., Davolio, A., Schiozer, D.J., 2020. Systematic approach to reduce uncertainties when quantitatively assimilating 4d seismic and well data. SPE Reserv. Eval. Eng. 23 (01), 1–12. http://dx.doi.org/10.2118/187081-PA.

Avseth, P., Mukerji, T., Mavko, G., 2010. Quantitative Seismic Interpretation: Applying Rock Physics Tools to Reduce Interpretation Risk. Cambridge Univ. Press, Cambridge, URL: http://cds.cern.ch/record/1601512.

Baddari, K., Ferahtia, J., Aifa, T., Djarfour, N., 2011. Seismic noise attenuation by means of an anisotropic non-linear diffusion filter. Comput. Geosci. 37 (4), 456–463. http://dx.doi.org/10.1016/j.cageo.2010.09.009.

Bonar, D., Sacchi, M., 2012. Denoising seismic data using the nonlocal means algorithm. Geophysics 77 (1), A5–A8. http://dx.doi.org/10.1190/geo2011-0235.1.

Brusdal, K., Brankart, J., Halberstadt, G., Evensen, G., Brasseur, P., van Leeuwen, P.J., Dombrowsky, E., Verron, J., 2003. An evaluation of ensemble based assimilation methods with a layered OGCM. J. Mar. Syst. 40–41, 253–289.

Canchumuni, S.W., Emerick, A.A., Pacheco, M.A.C., 2019. History matching geological facies models based on ensemble smoother and deep generative models. J. Pet. Sci. Eng. 177, 941–958. http://dx.doi.org/10.1016/j.petrol.2019.02.037, URL: http://www.sciencedirect.com/science/article/pii/S0920410519301743.

Chen, Y., Oliver, D.S., 2012. Ensemble randomized maximum likelihood method as an iterative ensemble smoother. Math. Geosci. 44, 1–26. http://dx.doi.org/10.1007/s11004-011-9376-z.

Chen, Y., Oliver, D.S., 2013. Levenberg-Marquardt forms of the iterative ensemble smoother for efficient history matching and uncertainty quantification. Comput. Geosci. 17, 689–703.

Chen, Y., Oliver, D.S., 2014. History matching of the norne full-field model with an iterative ensemble smoother. SPE Reserv. Eval. Eng. 17 (2), 244–256. http://dx.doi.org/10.2118/164902-PA.

Chen, Y., Oliver, D.S., 2017. Localization and regularization for iterative ensemble smoothers. Comput. Geosci. 21 (1), 13–30. http://dx.doi.org/10.1007/s10596-016-9599-7.

Coleou, T., Poupon, M., Azbel, K., 2012. Unsupervised seismic facies classification: A review and comparison of techniques and implementation. Lead. Edge 2, 942–953. http://dx.doi.org/10.1190/1.1623635.

Davolio, A., Schiozer, D.J., 2018. Probabilistic seismic history matching using binary images. J. Geophys. Eng. 15 (1), 261–274. http://dx.doi.org/10.1088/1742-2140/aa99f4.

Devegowda, D., Arroyo-Negrete, E., Datta-Gupta, A., Douma, S.G., 2007. Efficient and robust reservoir model updating using ensemble Kalman filter with sensitivity-based covariance localization. SPE 106144.

Donoho, D.L., Johnstone, I.M., 1995. Adapting to unknown smoothness via wavelet shrinkage. J. Amer. Statist. Assoc. 90 (432), 1200–1224. http://dx.doi.org/10.1080/01621459.1995.10476626.

Elad, M., Aharon, M., 2006. Image denoising via sparse and redundant representations over learned dictionaries. IEEE Trans. Image Process. 15 (12), http://dx.doi.org/10.1109/TIP.2006.881969.

Emerick, A.A., 2016. Analysis of the performance of ensemble-based assimilation of production and seismic data. J. Pet. Sci. Eng. 139, 219–239. http://dx.doi.org/10.1016/j.petrol.2016.01.029.

Emerick, A., Reynolds, A., 2011. Combining sensitivities and prior information for covariance localization in the ensemble Kalman filter for petroleum reservoir applications. Comput. Geosci. 15 (2), 251–269. http://dx.doi.org/10.1007/s10596-010-9198-y.

Emerick, A.A., Reynolds, A.C., 2013. Ensemble smoother with multiple data assimilation. Comput. Geosci. 55, 3–15.

Evensen, G., 1994. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. J. Geophys. Res. 99 (C5), 10,143–10,162.

Evensen, G., 2009a. Data Assimilation - The Ensemble Kalman Filter, second ed. Springer, p. 320. http://dx.doi.org/10.1007/978-3-642-03711-5.

Evensen, G., 2018. Analysis of iterative ensemble smoothers for solving inverse problems. Comput. Geosci. 22 (3), pp. 885–908. http://dx.doi.org/10.1007/s10596-018-9731-y.

Evensen, G., 2019. Accounting for model errors in iterative ensemble smoothers. Comput. Geosci. 23 (4), 761–775. http://dx.doi.org/10.1007/s10596-019-9819-z.

Evensen, G., Raanes, P., Stordal, A., Hove, J., 2019. Efficient implementation of an iterative ensemble smoother for data assimilation and reservoir history matching. Front. Appl. Math. Stat. 5, 47. http://dx.doi.org/10.3389/fams.2019.00047, URL: https://www.frontiersin.org/article/10.3389/fams.2019.00047.

Fahimuddin, A., Aanonsen, S.I., Skjervheim, J.-A., 2010. 4d seismic history matching of a real field case with enkf: Use of local analysis for model updating. SPE 134894.

Gaspari, G., Cohn, S.E., 1999. Construction of correlation functions in two and three dimensions. Q. J. R. Meteorol. Soc. 125, 723–757. http://dx.doi.org/10.1002/qj.49712555417.

Goda, T., Sato, K., 2014. History matching with iterative latin hypercube samplings and parameterization of reservoir heterogeneity. J. Pet. Sci. Eng. 114, 61–73. http://dx.doi.org/10.1016/j.petrol.2014.01.009, URL: http://www.sciencedirect.com/science/article/pii/S0920410514000217.

Gosselin, O., Aanonsen, S., Aavatsmark, I., Cominelli, A., Gonard, R., Kolasinski, M., Ferdinandi, F., Kovacic, L., Neylon, K., 2003. History matching using time-lapse seismic (HUTS). In: SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers, SPE-84464-MS.

Hamill, T.M., Whitaker, J.S., Snyder, C., 2001. Distance-dependent filtering of background error covariance estimates in an ensemble kalman filter. Mon. Weather Rev. 129 (11), 2776–2790. http://dx.doi.org/10.1175/1520-0493(2001)129<2776:DDFOBE>2.0.CO;2, arXiv:https://dx.doi.org/10.1175/1520-0493(2001)129<2776:DDFOBE>2.0.CO;2.

Han, J., van der Baan, M., 2015. Microseismic and seismic denoising via ensemble empirical mode decomposition and adaptive thresholding. Geophysics 80 (6), KS69–KS80. http://dx.doi.org/10.1190/geo2014-0423.1.

Hashin, Z., Shtrikman, S., 1951. Über die elastizität poröser medien. Viertel Nat.forsch. Ges Zrich 96, 1 – 23.

Hashin, Z., Shtrikman, S., 1963. A variational approach to the theory of the elastic behaviour of multiphase materials. J. Mech. Phys. Solids 11 (2), 127–140. http://dx.doi.org/10.1016/0022-5096(63)90060-7, URL: http://www.sciencedirect.com/science/article/pii/0022509663900607.

Helgerud, M.B., Miller, A.C., Johnston, D.H., Udoh, M.S., Jardine, B.G., Harris, C., Aubuchon, N., 2011. 4d in the deepwater gulf of Mexico: Hoover, madison, and marshall fields. Lead. Edge 30 (9), 1008–1018. http://dx.doi.org/10.1190/1.3640524.

Hennenfent, G., Herrmann, F.J., 2006. Seismic denoising with nonuniformly sampled curvelets. Comput. Sci. Eng. 8 (3), 16–25.

Johnston, D.H., 2013. Practical Applications of Time-Lapse Seismic Data, first ed. Society of Exploration Geophysicists, http://dx.doi.org/10.1190/1.9781560803126.

Kretz, V., Valles, B., Sonneland, L., 2004. Fluid front history matching using 4d seismic and streamline simulation. In: SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers, http://dx.doi.org/10.2118/90136-MS, SPE-90136-MS.

Latif, A., Mousa, W.A., 2017. An efficient undersampled high-resolution radon transform for exploration seismic data processing. IEEE Trans. Geosci. Remote Sens. 55 (2), 1010–1024.

van Leeuwen, P.J., Evensen, G., 1996. Data assimilation and inverse methods in terms of a probabilistic formulation. Mon. Weather Rev. 124, 2898–2913.

Leeuwenburgh, O., Arts, R., 2014. Distance parameterization for efficient seismic history matching with the ensemble kalman filter. Comput. Geosci. 18, 535–548. http://dx.doi.org/10.1007/s10596-014-9434-y.

Liu, M., Grana, D., 2020. Time-lapse seismic history matching with iterative ensemble smoother and deep convolutional autoencoder. Geophysics 85 (1), M15–M31. http://dx.doi.org/10.1190/geo2019-0019.1.

Liu, E., Jafarpour, B., 2013. Learning sparse geologic dictionaries from low-rank representations of facies connectivity for flow model calibration. Water Resour. Res. 49, 7088–7101. http://dx.doi.org/10.1002/wrcr.20545.

Lorentzen, R., Bhakta, T., Grana, D., Luo, X., Valestrand, R., Nævdal, G., 2020. Simultaneous assimilation of production and seismic data: Application to the Norne field. Comput. Geosci. 24, 907–920. http://dx.doi.org/10.1007/s10596-019-09900-0.

Lorentzen, R., Luo, X., Bhakta, T., Valestrand, R., 2019. History matching the full Norne field model using seismic and production data. SPE J. 24, 1452–1467. http://dx.doi.org/10.2118/194205-PA, SPE-194205-PA.

Luo, X., 2019. Ensemble-based kernel learning for a class of data assimilation problems with imperfect forward simulators. PLOS ONE 14 (7), 1–40. http://dx.doi.org/10.1371/journal.pone.0219247.

Luo, X., Bhakta, T., 2020. Automatic and adaptive localization for ensemble-based history matching. J. Pet. Sci. Eng. 184, 106559. http://dx.doi.org/10.1016/j.petrol.2019.106559.

Luo, X., Bhakta, T., Jakobsen, M., Nævdal, G., 2017. An ensemble 4D-seismic history-matching framework with sparse representation based on wavelet multiresolution analysis. SPE J. 22, 985–1010. http://dx.doi.org/10.2118/180025-PA, SPE-180025-PA.

Luo, X., Bhakta, T., Jakobsen, M., Nævdal, G., 2018. Efficient big data assimilation through sparse representation: A 3D benchmark case study in petroleum engineering. PLOS ONE 13 (7), e0198586. http://dx.doi.org/10.1371/journal.pone.0198586.

Luo, X., Lorentzen, R.J., Valestrand, R., Evensen, G., 2019. Correlation-based adaptive localization for ensemble-based history matching: Applied to the Norne field case study. SPE Reserv. Eval. Eng. 22, 1084–1109. http://dx.doi.org/10.2118/191305-PA, SPE-191305-PA.

Luo, X., Stordal, A., Lorentzen, R., Nævdal, G., 2015. Iterative ensemble smoother as an approximate solution to a regularized minimum-average-cost problem: theory and applications. SPE J. 20, 962–982. http://dx.doi.org/10.2118/176023-PA, SPE-176023-PA.

Maschio, C., Schiozer, D.J., 2016. Probabilistic history matching using discrete Latin Hypercube sampling and nonparametric density estimation. J. Pet. Sci. Eng. 147, 98–115. http://dx.doi.org/10.1016/j.petrol.2016.05.011.

Mavko, G., Mukerji, T., Dvorkin, J., 2009. The Rock Physics Handbook: Tools for Seismic Analysis of Porous Media, second ed. Cambridge University Press, http://dx.doi.org/10.1017/CBO9780511626753.

Mindlin, R.D., 1949. Compliance of elastic bodies in contact. J. Appl. Mech. ASME 16, 259–268, URL: https://ci.nii.ac.jp/naid/10013415139/en/.

Nævdal, G., Mannseth, T., Vefring, E.H., 2002. Near-Well reservoir monitoring through ensemble Kalman Filter. SPE Int. 75235 (1), 1–9. http://dx.doi.org/10.2118/75235-MS.

Obidegwu, D., Chassagne, R., MacBeth, C., 2017. Seismic assisted history matching using binary maps. J. Nat. Gas Sci. Eng. 42, 69–84. http://dx.doi.org/10.1016/j.jngse.2017.03.001, URL: http://www.sciencedirect.com/science/article/pii/S187551001730104X.

Oliver, D.S., Chen, Y., 2011. Recent progress on reservoir history matching: A review. Comput. Geosci. 15 (1), 185–221. http://dx.doi.org/10.1007/s10596-010-9194-2.

Peters, L., Arts, R., Brouwer, G., Geel, C., Cullick, S., Lorentzen, R.J., Chen, Y., Dunlop, N., Vossepoel, F.C., Xu, R., Sarma, P., Alhuthali, A.H., Reynolds, A., 2010. Results of the Brugge Benchmark study for flooding optimization and history matching. SPE Reserv. Eval. Eng. http://dx.doi.org/10.2118/119094-PA.

Raanes, P.N., Stordal, A.S., Evensen, G., 2019. Revising the stochastic iterative ensemble smoother. Nonlinear Process. Geophys. 26, 325–338. http://dx.doi.org/10.5194/npg-2019-10, URL: https://doi.org/10.5194/npg-26-325-2019.

Ravishankar, S., Bresler, Y., 2011. Mr image reconstruction from highly undersampled k-space data by dictionary learning. IEEE Trans. Med. Imaging 30 (5), 1028–1041. http://dx.doi.org/10.1109/TMI.2010.2090538.

Roggero, F., Lerat, O., Ding, D.Y., Berthet, P., Bordenave, C., Lefeuvre, F., Perfetti, P., 2012. History matching of production and 4d seismic data: Application to the girassol field, offshore angola. Oil Gas Sci. Technol. - Rev. IFP Energies Nouv. 67 (2), 237–262. http://dx.doi.org/10.2516/ogst/2011148.

Rubinstein, R., Zibulevsky, M., Elad, M., 2008. Efficient implementation of the K-SVD algorithm using Batch Orthogonal matching pursuit. Technical Report, Computer Science Department, Technion - Israel Institute of Technology, URL: http://www.cs.technion.ac.il/{~}ronrubin/software.html..

Sakov, P., Bertino, L., 2011. Relation between two common localization methods for the EnKF. Comput. Geosci. 15, 225–237.

Shan, H., Ma, J., Yang, H., 2009. Comparisons of wavelets, contourlets and curvelets in seismic denoising. J. Appl. Geophys. 69 (2), 103–115. http://dx.doi.org/10.1016/j.jappgeo.2009.08.002.

Skjervheim, J.-A., Evensen, G., Hove, J., Vabø, J., 2011. An ensemble smoother for assisted history matching. SPE 141929.

Soares, R., Luo, X., Evensen, G., 2019. Sparse representation of 4D Seismic signal based on dictionary learning. In: SPE Norway One Day Seminar. http://dx.doi.org/10.2118/195599-MS.

Soares, R.V., Maschio, C., Schiozer, D.J., 2018. Applying a localization technique to kalman gain and assessing the influence on the variability of models in history matching. J. Pet. Sci. Eng. 169, 110–125. http://dx.doi.org/10.1016/j.petrol.2018.05.059, URL: http://www.sciencedirect.com/science/article/pii/S0920410518304534.

Turquais, P., 2018. Dictionary Learning and Sparse Representations for Denoising and Reconstruction of Marine Seismic Data (Ph.D. thesis). University of Oslo, Oslo, p. 145.

Xiong, M., Li, Y., Wu, N., 2014. Random-noise attenuation for seismic data by local parallel radial-trace TFPF. IEEE Trans. Geosci. Remote Sens. 52 (7), 4025–4031.

Zhang, Q., Li, B., 2010. Discriminative k-SVD for dictionary learning in face recognition. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 2691–2698. http://dx.doi.org/10.1109/CVPR.2010.5539989.

Zhu, W., Mousavi, S.M., Beroza, G.C., 2019. Seismic signal denoising and decomposition using deep neural networks. IEEE Trans. Geosci. Remote Sens. 57 (11), 9476–9488.