Multiscale Ensemble Filtering in Reservoir Engineering Applications

Wiktoria Lawniczak

Master's Thesis

Technical University in Delft

SUPERVISORS

Prof. dr ir Arnold W. Heemink

Dr Remus Hanea

Prof. Dennis McLaughlin (MIT)

Committee members

Prof. dr. ir A. W. Heemink	EEMCS, DIAM, TUDelft
Prof. dr. ir J. D. Jansen	CITG, TUDelft
Dr. R. G. Hanea	TNO, Utrecht
Dr. C. B. M. Stroet	TNO, Utrecht
Dr. D. Kurowicka	EEMCS, DIAM, TUDelft

ACKNOWLEDGEMENT

This thesis was sponsored by TNO and was considered a part of the ISAPP program. I would like to thank Chris Stroet and Jan Dirk Jansen for the given opportunities.

It would not be possible for me to begin and finish these studies if it was not for Prof. Jolanta Misiewicz, Prof. Roger Cooke and Dr. Dorota Kurowicka. Thank you for this chance.

I am very grateful to Prof. Dennis McLaughlin and Prof. Arnold Heemink for their supervision, ideas and help.

Thank you, Crystal and Behnam, for your help in the very beginning of my research. I could not have started it without you.

I would like to thank Frank who was so kind to spend his time explaining me the basics of reservoir engineering.

It was very nice to have an opportunity to talk to friends interested in my research. Thank you, Anca and Oswaldo.

Especially, I would like to thank Remus who is not only a patient supervisor but also a good friend. He has been by my side all that time and helped some dreams come true.

I would like to thank my friends and family, they made me very lucky to be surrounded by those who believe in me. Mom and Dad, thank you so much for everything. The best office mates anybody could ever have are dearest friends to me, thank you, Angela and Marcin. A group of friends who never forget about me: Gosia, Ola, Karolina and Asia; thank you. And last but very special, thank you, Carlos.

Contents

1 Introduction

In the field of reservoir engineering new methods of assimilating the data are needed because of specific types of observations. They can be obtained from several sources and differ from the measurements in other types of applications. The secondary recovery is considered where water flooding is performed. The typical is five point observation network when the wells are drilled in the corners of the field and its middle. This small data set can be enriched with large scale data, for instance, from the seismic or satellite observations. This very noisy data is expensive in assimilation if one uses ensemble Kalman filtering (EnKF).

The aim of the data assimilation in reservoir engineering is to predict subsurface characteristics which give an idea about the flow of the fluids through the reservoir (permeability). The four wells that are in the corners of the field are the producers, the middle one is an injector. The water is pumped through the injector to push the oil up the producers. During that process measurements are taken. It is not possible to observe permeability, therefore, the measurements are flow rates (differences in pressure). Those values, using the dependencies in the model simulator, are translated into the permeability.

Given the observations, one tries to predict the ability of the fluid flow in the reservoir (permeability). A new algorithm - ensemble multiscale filter (EnMSF) - was proposed. This work points out some problems met so far in reservoir engineering area and checks how well they can be solved by applying EnMSF. Its roots come from image processing and it is expected to overcome EnKF's disadvantages. The ensemble Kalman filter tends to preserve the correlations between physically independent points and takes a lot of time assimilating large scale data. It is expected to be worked out by the new algorithm.

It became important to focus on the properties of ensemble multiscale filter and test it. Since the algorithm is new, it is also important to assess its performance, check the properties and learn how to use it efficiently. There are several parameters in the algorithm which so far depend on the choice of a user. This flexibility of the method can be seen as an advantage but also as a disadvantage. It can help in adjusting the algorithm to various applications but, on the other hand, it might be hard to control all the parameters.

The thesis is organized in the following way. Section 2 introduces the information about the ensemble multiscale algorithm, the equations and assumptions. The next two sections, 3 and 4, contain some examples. The first one is more theoretical where the measurements are permeability values and can be constructed in an arbitrary way. It shows how the filter deals with different kind of data. The second one comes from the real-life application and shows how changing some parameters in the algorithm can influence its performance, given the measurements and the ensemble. Therefore, the theoretical example focuses more on the choice of the measurements, the practical one more on the tree topology. Both examples are going to stress and test some of the features of the ensemble multiscale filter. Everything is followed by conclusions in Section 5.

2 Short introduction to multiscale ensemble Kalman filter

To introduce the multiscale filter some basics of Kalman filtering ([5]) have to be shown. Some information about the Kalman filter will help to get acquainted with ensemble version of the algorithm what is the prior knowledge to EnMSF.

Both, Kalman filter and EnKF, provide a way to perform a forecast and update of the states. In the forecast step the state is propagated in time. So, having the state at time t it predicts its value at time $t + 1$. In the update step, model predictions obtained in the forecast step are combined with the data from the measurements to get an optimal estimate.

Kalman filter begins with the initial value of the states and their initial covariance matrix. It propagates the covariance and trough a Kalman gain (weighting factor) updates the states with measurements (see Appendix C for equations). In ensemble Kalman filter this initial knowledge of the state is expressed via an ensemble which represents the distribution of the truth. There the update is performed on each ensemble member again with a Kalman gain being a weighting factor (see Appendix D for equations).

Kalman filters are known for keeping the long distance correlations and computational problems if the matrices representing the states and measurements are large. In reservoir engineering the fields of states are spanned over a huge area and physically points which are far away in reality should not be correlated. EnMSF gets rid of long distance correlation by prioritizing high dependencies. Also, working with those small covariances allows to make faster the computations with large scale data.

NOTATION

s	Nodes on the tree.
$s\alpha_i$	Child i of node s.
$s\gamma$	Parent of node s.
$\chi(s)$	State vector at node s.
$\chi_M(s)$	The vector of finest-scale states descended from s.
$m(s)$	Scale of node s.
$n(s)$	Dimension of the state vector $\chi(s)$.
$z_i(s)$	Vector of states at node $s\alpha_i$; $\chi(s\alpha_i)$.
$z_{ic}(s)$	Vector of the states on all nodes at scale $m(s)+1$ except node $s\alpha_i$.
$y(s)$	Measurement vector in node s.
$h(s)$	Measurement matrix in node s.
$e(s)$	Local measurement error vector with 0 mean and covariance r(s).
$\chi(s\|s)$	State update at node s.
$\chi(s\|S)$	Smoothed state at node s.
$\chi(s\gamma\|s)$	Projected state at node $s\gamma$.
^{j}	Superscript indicating an ensemble.

In general, Kalman filter methods consist of two main steps: the forecast and the update but the multiscale ensemble Kalman filter[1] provides a way to perform the update step only. Like in EnKF the computations are done on the ensemble.

It consists of three basic steps:

1. assigning grid cells (pixels) to the finest scale nodes and describing the tree parameters from sample propagation through the tree (tree construction);

2. upward sweep (update);

3. downward sweep (smoothing).

Given the ensemble, the Multiscale algorithm constructs a proper tree, places the ensemble members on the finest scale and propagates them up and down the tree. Then the output is a set of updated and smoothed replicates.

The most complex is step 1., containing crucial assumptions but also many flexible variables. Identifying appropriate tree parameters is easier assuming *locally internal model*. It means that the state at each non-fine-

[1]Section based on [1]

scale node is a linear combination of the states at its children:

$$\chi(s) = V(s) \begin{bmatrix} \chi(s\alpha_1) \\ \vdots \\ \chi(s\alpha_q) \end{bmatrix}.$$

Given matrices $V(s)$ one can perform the upward and downward recursion with:

Downward recursion equation

$$\chi(s) = A(s)\chi(s\gamma) + w(s),$$

Upward recursion equation

$$\chi(s\gamma) = F(s)\chi(s) + w'(s),$$

where $w(s)$ and $w'(s)$ are zero-mean random scale perturbation with covariances $Q(s)$, $Q'(s)$, respectively. Specifying matrices $V(s)$ is equivalent to specifying $A(s)$, $F(s)$, $Q(s)$ and $Q'(s)$ (Appendix A) from upward and downward recursion equations.

We search for a set of $V(s)$'s that provides *the scale-recursive Markov property* on the tree, i. e. decorrelates $q+1$ following sets: first q sets are all the children of the node s, and the set $q+1$ contains all the nodes of scale $m(s)+1$ that are not children of s.

The tree that will approximate the forecast covariance matrix well should be based on *the scale-recursive Markov property.* The set of $V(s)$'s providing the scale-recursive Markov property perfectly would have a very high dimension since it would keep the total dependence between the finest states on the upper scale. Therefore, for practical purpose the state dimensions in coarser scales will be constrained. This is easier if $V(s)$'s are *block diagonal*; each block corresponds to a different and only one child of s.

The way $V(s)$'s are built

$V(s)$ has the form:

$$V(s) = diag[V_1(s), ..., V_q(s)],$$

where $V_i(s)$ is a matrix corresponding to the ith child of s, $s\alpha_i$, for $i = 1, ..., q$.

There are two constraints hidden here. The first one limits the number of rows in matrices $V_i(s)$ to $d_i(s) < n(s\alpha_i)$. The second one, if necessary, coarsens the number of rows in $V(s)$ to $d(s)$. In both cases, the appropriate choice of the rows is based on an eigenvalue decomposition.

Constructing matrices $V_i(s)$

To obtain $V_i(s)$'s, q conditional covariances would have to be minimized, for each non-fine-scale node s. Those would be the conditional cross-covariances between child i ($i = 1, ..., q$) and the rest of the nodes in the same scale, given the parent. Since direct minimization is inconvenient, the algorithm uses a *predictive efficiency method.*

Predictive efficiency method

The method is more efficient to compute than all the conditional cross-covariances. It picks $V_i(s)$'s which minimize the departure from optimality of the estimate:

$$\hat{z}_{ic}(s) = E[z_{ic}(s)|V_i(s)z_i(s)],$$

where $z_i(s)$ is a vector of states at node $s\alpha_i$ $(= \chi(s))$ and $z_{ic}(s)$ is a vector of states on all nodes at scale $m(s)+1$ except node $s\alpha_i$. It was proved that they are given by the first $d_i(s)$ rows of:

$$V_i'(s) = U_i^T(s)Cov[z_i(s)]^{-1/2},$$

where $U_i(s)$ contains the column eigenvectors of:

$$Cov^{-1/2}[z_i(s)]Cov[z_i(s), z_{ic}(s)]Cov^T[z_i(s), z_{ic}(s)]Cov^{-T/2}[z_i(s)]. \qquad (1)$$

Here it should be noted that $d_i(s)$ are chosen by the user. The picked rows have the highest corresponding eigenvalues. The reason is that we assume that the column eigenvectors of $U_i(s)$ are lined in a decreasing (corresponding eigenvalue) order.

It is important to notice that while coding the algorithm, sample covariances $\widehat{Cov}$ in (1) are used. Computing those cross-covariances might be demanding since $z_{ic}(s)$ can be large. Here the notion of *neighborhood* is introduced. It chooses only states on the nodes in the range of given radius from the node $s\alpha_i$. Those states only are included in the complementary vector $z_{ic}(s)$. At this point it becomes visible how important it is to wisely define the *grid-node transformation*.

Grid-node transformation

It is all about assigning the 'real life' grid cells to the finest-scale nodes of the tree. Again, all depends on the user. The important thing is to preserve the 'real life' physical dependence between cells in the tree. Then the neighborhood contains the closest, most influential points.

Additional task is to pick the structure of the tree, i. e. the number of children in every node which may vary from parent to parent.

After building the tree and getting all the needed parameters using the method described above, it is possible to perform the update of the ensemble with all the given measurements. Multiscale algorithm allows to put measurements in the higher scales as well as in the finest one.

It is important to notice that any measurement depends only on the finest-scale states:

$$y(s) = h(s)\chi_M(s) + e(s),$$

where $e(s)$ is zero-mean with covariance $r(s)$. It is assumed that $h(s)$ and covariance matrix $r(s)$ are known for each node with a measurement.

Going up the tree the algorithm updates the states on the nodes. Then each node s gets the value $\chi^j(s|s)$. $\chi^j(s|s)$ is the state vector updated with all the measurements in the subtree rooted at s.

At the top of the tree the value for the root node is obtained, $\chi^j(0|0)$. This is the basis to perform the downward sweep of the algorithm. $\chi^j(0|0)$ is the initial point, namely $\chi^j(0|S)$, for the smoothing. Going down the tree the value $\chi^j(s|S)$ is assigned to each node s. That is the smoothed state value containing the knowledge from all given measurements. This way at the end of the sweep we get smoothed updated states at the finest scale which can be used to perform the next forecast step.

Additional general equations leading the update and the smoothing[2]

The upward sweep equation

$$\chi^j(s|s) = \chi^j(s) + K(s)[Y^j(s) - \hat{Y}^j(s)]$$

is very similar to the traditional Kalman filter equation. The states $\chi^j(s)$ on each node are updated with perturbed measurements $Y^j(s)$ using weighting factor $K(s)$ and predicted measurements $\hat{Y}^j(s)$ (see Appendix B).

[2]All details can be found in [1] and some equations in Appendix B

The downward sweep equation

$$\chi^j(s|S) = \chi^j(s|s) + J(s)[\chi^j(s\gamma|S) - \chi^j(s\gamma|s)]$$

for $m(s) > 0$ is based on the RTS smoother equation, where $J(s)$ is the smoothing gain and $\chi^j(s\gamma|S)$ is initially known from the upward sweep (for $s = 0$). It is important to mention that to compute projected replicates $\chi^j(s\gamma|s)$ upward recursion equation is used in the form:

$$\chi^j(s\gamma|s) = F(s)\chi^j(s|s) + w'^j(s).$$

2.1 Theoretical assumptions

The algorithm is based on two important assumptions[3]: local internality and Markovianity. The model is internal when each state at coarser scale is a linear combination of its children. It makes model construction easier and allows using multiresolution measurements by putting them higher in the tree structure which is believed to make the algorithm computationally attractive and overcomes computational burdens of previous approaches. Additionally, assuming internality, the Markov property and scale-recursive Markov property are equivalent. The latter one is of interest for this algorithm. It stands that the following q+1 sets are independent, given node s: each of q children

[3]This subsection is greatly based on [3] where the meaning and importance of Markovianity and internality is described in details.

of s, and the set containing all other nodes from the same scale as the children. Scale-recursive Markov property provides more efficient computation of the statistics (covariances). The Markov property says that sets of nodes in $q + 1$ trees created by removing node s are conditionally uncorrelated, given node s.

There are several less strict assumptions used in the multiscale algorithm. Mostly, they introduce practical and computational simplifications. Their short specification is given below.

Constraining the dimensions of matrices V(s) (It chooses only the most influential grid points. Therefore, it chooses the essence needed to perform the computations correctly. This way it cuts off the long distance or weakly correlated points.)

Block diagonal structure of V(s) (Each block corresponds to a different and only one child of s.)

Predictive efficiency method (It is used since the computation of all cross-covariances would be expensive to perform.)

The features of the algorithm which entirely depend on the user are:

- number of pixels per finest-scale node,
- number of children per parent,
- numbering of the grid cells,

- the neighborhood radius,
- dimension constraints on matrices $V(s)$ and $V_i(s)$.

Their influence is tested in further sections.

The multiscale algorithm has a potential to be a good assimilation tool for reservoir engineering applications. It keeps small dimensions of covariances on the higher scales what makes it not expensive computationally. The measurement inclusion is introduced in a new efficient way for large scale data. It is believed that, thanks to many user-dependent parameters, it can be adjusted properly to the problem using all available data.

3 Theoretical example

First the algorithm is run with a theoretical example which allows to test more features than the practical example shown further.

Given the training image (Figure 1) the ensemble of 1001 members was generated using SGeMS (The Stanford Geostatistical Modeling Software). All samples are 2D permeability fields of size 64×64. The first replicate was assumed to be the 'truth' (Figure 2) and removed from the ensemble.

This example allows for testing various kinds of measurements on the finest as well as on the coarser scales. The behavior of the filter given different types of data is tested.

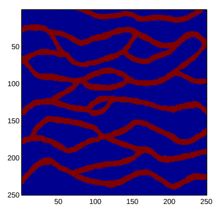

Figure 1: Training image

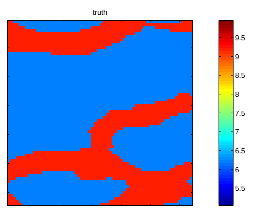

Figure 2: Truth for the theoretical example

The values of the observations are the perturbed values of the 'truth'. It means that the permeability field is updated with permeability measurements. In practice these values cannot be measured. Therefore, the theoretical example is not realistic but allows to test a wide variety of cases and possibilities.

Throughout the tests the tree is a quadtree, there are 16 pixels assigned to each finest-scale node, 16 states preserved at coarser scales and no restriction on the neighborhood radius. Only one numbering scheme is used:

$$\begin{array}{|cccc|ccc}
\hline
1 & 2 & 5 & 6 & 17 & 18 & \cdots \\
3 & 4 & 7 & 8 & 19 & 20 & \\
9 & 10 & 13 & 14 & & & \\
11 & 12 & 15 & 16 & & & \\
\cline{1-4}
\vdots & & & & & &
\end{array}.$$

The task in this section is to analyze the usefulness of different large scale measurement fields. This kind of data might be obtained from a satellite or seismic tests.

The table below contains the root-mean-square errors between the truth and the prior computed from the ensemble of given size. The mean of the ensemble should be the best estimate that can be got if there are no measurements. Given the error of the prior it can be checked if the filter, using the measurements, improved the estimation.

Ens. size	RMSE
10	1.4719
20	1.3883
30	1.4053
40	1.3540
50	1.3789
60	1.3820
70	1.3894
80	1.3956
90	1.3966
100	1.4002

3.1 Measurements taken per pixel

Two measurement areas are assumed: Figure 3 and Figure 4, both with standard deviation of the error equal to 0.001. The fields are of size 16×16, the first one contains a channel but the second one does not. For each case the performance of the filter is assessed when the measurements are put on the finest (fourth) or the second scale of the tree. It will be shown if EnKF or EnMSF can track the channels and how well they perform.

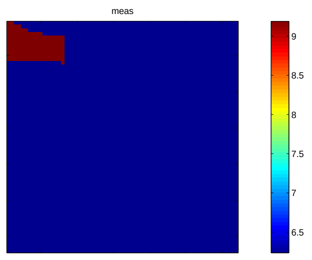

Figure 3: Measurement over a large area with a channel

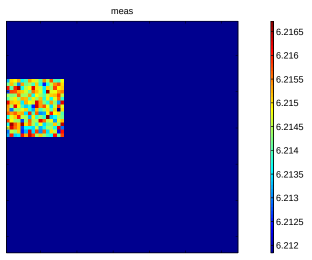

Figure 4: Measurement over a large area without a channel

The observations are done in every pixel. Therefore, assisting matrices are large and can be computationally demanding. It is tested if the same data placed higher in the tree has any impact on the filter's performance.

The tables with measurements placed on the finest scale are organized as follows. The first column is the number of ensemble members. Usually the experiment is done for ensembles with at most 50 members. The results for greater ensembles are shown if they are significant. The second column is the time which EnMSF takes to construct the tree. The next two columns are the times of update in EnMSF and EnKF, respectively. All times are shown in seconds. The next-to-last and last columns contain the root-mean-square errors of EnKF and EnMSF, respectively. When the measurements are placed higher in the tree only EnMSF's results are available.

The first thing to notice in the tables is the long time needed for the tree identification which grows with the size of the ensemble. The results shown here were obtained by running the algorithm with no restrictions on the neighborhood radius. Reducing the radius does not influence the RMS

Ens. size	Tree id.[s]	EnMSF time[s]	EnKF time[s]	RMSE EnMSF	RMSE EnKF
10	2.6094	0.4375	10.6719	1.5847	2.0654
20	4.5156	0.39063	10.8281	1.4356	1.9298
30	6.7969	0.59375	11.9375	1.3916	1.706
40	8.1719	0.46875	11.25	1.4619	1.5878
50	11.5469	0.59375	11.5	1.2704	1.6769
60	13.9375	0.60938	11.5156	1.3104	1.8355
70	15.7188	0.625	11.7969	1.4603	2.0175

Table 1: Observation with a channel placed on the finest scale

Ens. size	Tree id.[s]	EnMSF time[s]	RMSE EnMSF
10	2.5156	0.95313	3.012
20	4.5938	1	1.7456
30	6.2656	1.1875	1.8174
40	8.0938	1.0156	1.7347
50	11.2188	1.1875	2.0515
60	13.6406	1	1.686
70	15.5	1.0156	1.8271
80	17.6094	1	1.5365
90	19.3906	1.1875	1.4627
100	21	1.0625	1.4787

Table 2: Observation with a channel placed on the upper scale

error and decreases the tree identification time (this issue is further discussed in Section 4). Additionally, in comparison to the computational time of any forward simulator, this range of values is very small.

The time of update in EnMSF, given the tree parameters, is around twice as large when the measurements are put up the tree than on the finest scale. This happens due to the character of the observation. Since its value is known in every pixel, the matrices on the higher scale covering the whole

Ens. size	Tree id.[s]	EnMSF time[s]	EnKF time[s]	RMSE EnMSF	RMSE EnKF
10	2.3906	0.35938	10.8125	1.4105	1.7019
20	4.7813	0.5	10.8438	1.3287	1.4334
30	6.1719	0.42188	11.1563	1.3756	1.3704
40	8.3281	0.57813	11.3125	1.3197	1.3142
50	11.2188	0.48438	11.3281	1.3256	1.3293
60	13.625	0.5	11.5156	1.3366	1.3351
70	15.5313	0.625	11.6406	1.3604	1.3453

Table 3: Observation without a channel placed on the finest scale

Ens. size	Tree id.[s]	EnMSF time[s]	RMSE EnMSF
10	2.7188	1.1563	1.5541
20	4.5469	0.98438	1.4071
30	6.4688	1.1875	1.3915
40	8.2813	1.0469	1.3588
50	12.1563	1.0469	1.357
60	13.7188	1.0469	1.3771

Table 4: Observation without a channel placed on the upper scale

area are larger what causes the increase of computational time. The time of EnKF update is quite large and increases slowly with the ensemble size. Since the measurement size is large and heavy for EnKF, the number of ensemble members becomes a minor problem.

From the information on RMS error it can be concluded that the observation without a channel is in general not very informative. It gives similar results for all options but in the same time in comparison to prior estimation it is better in every case.

An interesting observation is that for the measurement with a channel

EnKF tends to diverge. Its RMSE becomes gradually worse with the number of ensemble members what is already visible for ensemble size equal to 70 (Figure 5). The top plot is the prior of the ensemble. In the lower row: EnKF update on the left and EnMSF update on the right. In this case the best estimate is obtained for 50 ensemble members where the measurement is placed on the finest scale (Figure 6). This estimation is also better than the prior.

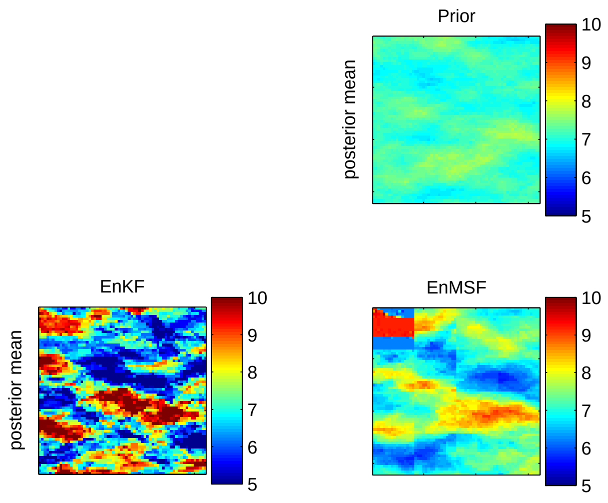

Figure 5: Theoretical example. The results of assimilating large scale data (in upper left corner). Top: prior. Lower left: EnKF. Lower right: EnMSF

In general, placing the observations of this type on the coarser or finer scales of EnMSF does not show significant difference. The purpose of this subsection was to introduce various possibilities of looking at the same observations. Putting them on the coarser scales allows to use measurements

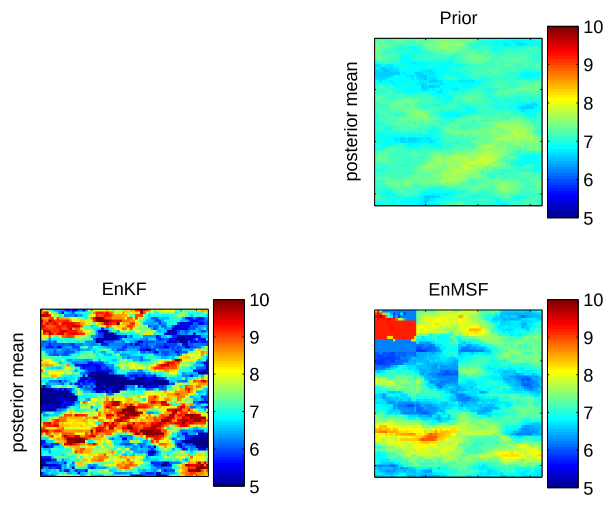

Figure 6: Theoretical example. The best result for assimilating the data with a channel using 50 replicates. The data is given for every pixel and placed on the finest scale

from different sources in one run.

3.2 Measurements as a perturbed mean

A different set of observations is constructed. Here it is assumed that the perturbed mean of the values over some area is available. As before the upper left square of size 16×16 is measured but three scales are investigated. The data is placed on sixteen nodes from the finest scale (Figure 7), four nodes one scale higher (Figure 8) or one node two scales higher than the finest scales (Figure 9). All three options span the same measurement area. It is checked how beneficial is placing this type of observations on the coarser scales and how well they can be used.

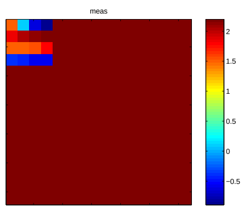

Figure 7: Theoretical example. Measurement of the mean placed on 16 finest scale nodes

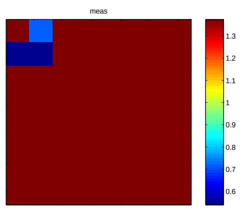

Figure 8: Theoretical example. Measurement of the mean placed on 4 coarser scale nodes

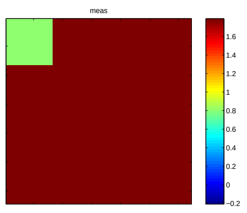

Figure 9: Theoretical example. Measurement of the mean placed on one node covering the area

The tables are organized as before. Each one refers to a different placement of the measurement on the tree and reaches up to 50 ensemble members.

The time of building the tree is comparable for all three experiments and, as before, could be reduced. The time of update reduces with scale coarsening. This is due to the number of updates that need to be done. For the measurements placed on the coarser scales less nodes span the same measurement area. It does not make the filter work worse. In this example it might be insignificant but for large scale examples it will have an importance especially with a large number of data. Additionally, the results seem to improve

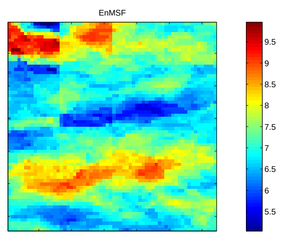

Figure 10: Theoretical example. The assimilation of the mean measurement for 40 ensemble members, placed on the finest scale

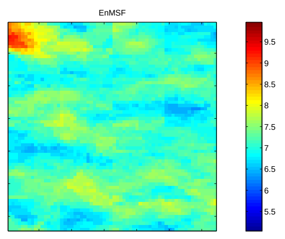

Figure 11: Theoretical example. The assimilation of the mean measurement for 40 ensemble members, placed on the scale **one** up from the finest

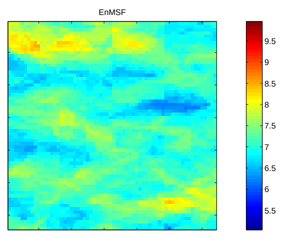

Figure 12: Theoretical example. The assimilation of the mean measurement for 40 ensemble members, placed on the scale **two** up from the finest

with higher tree scales. For an example the plots for 40 ensemble members are shown. Figure 10 shows the result for the finest scale. Figures 11 and 12 show the results for one and two scales higher, respectively. The second scale result appears very similar to the prior, therefore the finest scale estimation seems to be more informative with its sharper results. Measurement placed on the third scale give the result in between the other two capturing the features in the upper left corner. The two higher scales measurements give

Ens. size	Tree id.[s]	EnMSF time[s]	RMSE EnMSF
10	2.875	0.5	2.1354
20	5.3281	0.5	2.5813
30	6.8125	0.67188	1.9983
40	9.0313	0.67188	1.3352
50	12.3438	0.73438	1.4341

Table 5: Observation of the mean placed on the finest scale (fourth)

Ens. size	Tree id.[s]	EnMSF time[s]	RMSE EnMSF
10	2.9219	0.46875	1.4806
20	5.375	0.46875	1.342
30	6.7969	0.625	1.3347
40	9.0469	0.65625	1.3471
50	12.1406	0.67188	1.3322

Table 6: Observation of the mean placed on the third scale

better estimations than the prior, whereas the finest scale is worse. Nevertheless, the last two plots more and more resemble the prior. It cannot be avoided if only the assimilation is performed without any forward model to perform long run predictions.

Ens. size	Tree id.[s]	EnMSF time[s]	RMSE EnMSF
10	3.3438	0.48438	1.3633
20	5.2656	0.5625	1.294
30	6.7813	0.60938	1.3168
40	9.0156	0.65625	1.303
50	12.0938	0.65625	1.3094

Table 7: Observation of the mean placed on the second scale

3.3 Measurement of the whole field

Very often it might be possible to obtain the measurement in every pixel of the field. This kind of data is very noisy and obviously very large. It is known that EnKF is not an efficient tool to assimilate observations of this size.

Here, the observation is the true field perturbed by noise with standard deviation equal to 9 (Figure 13).

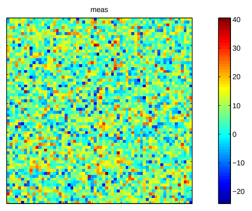

Figure 13: Very noisy measurement of the whole field

From the table it can be seen how time consuming EnKF is in comparison to EnMSF. As in previous example, the time of EnKF does not grow rapidly with ensemble size. The size of the measurement puts much heavier computational load on the filter than the ensemble size. Its RMS error does

Ens. size	Tree id.[s]	EnMSF time[s]	EnKF time[s]	RMSE EnMSF	RMSE EnKF
10	2.8438	1.2188	582.875	1.273	1.4075
20	5.1719	1.4844	582.4531	1.158	1.2978
30	6.3906	1.5625	584.9844	1.1868	1.3629
40	8.8438	1.6719	585.6875	1.1297	1.174
50	11.75	1.7656	586.0625	1.0745	1.1912
60	14.0156	1.9219	587.1406	1.0685	1.186
70	15.8125	1.9375	587.7813	1.0959	1.1831

Table 8: Observation of the whole field

not give good results neither. Whereas, EnMSF can already give informative results for ensemble as small as 20. Just for the example the plots for 50 replicates are shown (Figure 14).

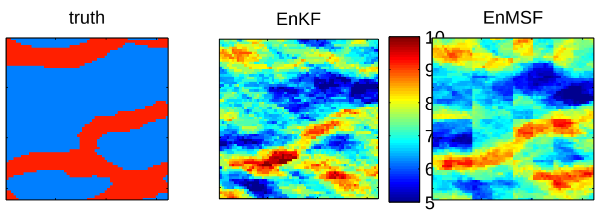

Figure 14: The truth and results for assimilating the observation of the whole field using 50 replicates. The middle: EnKF. The right hand side: EnMSF

On the left hand side there is the truth, in the middle EnKF, and EnMSF on the right hand side. Although the EnMSF shows the square pattern, it is locally better than EnKF catching the important features.

This Section shows that EnMSF can deal well with large scale measurements and it is not that sensitive on the size of the ensemble. It gives good results in a reasonable time.

4 Practical example

Previous trials were done with replicates of permeability fields generated using SGeMS. They were fields of the size 64x64 pixels (cells). The code of the algorithm is not yet universal. Generally speaking, it was written for fields with size equal to a power of two and trees with the same number of children for each coarser-scale node.

It has become possible to use 94 replicates[4] of size 48×48 generated using Mejia's algorithm which were later forecasted with MoRes simulator. The replicates are the possible permeability field realizations in 2D which mean is shown in Figure 17. The measurements of flow rates were taken in five points: each corner of the field and the middle (Figure 15). The set of five forecasted measurement points is given per each ensemble member. The true permeability field is also available (Figure 16). In [4] the ensemble was used to perform EnKF which results will be used for comparison (Figure 18).

Since the size of the replicates is 48x48 it was necessary to work more

[4]This section is based on [4].

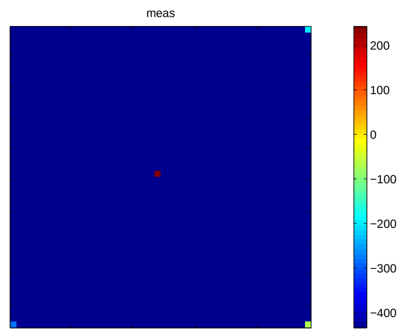

Figure 15: The five point measurement for the practical example

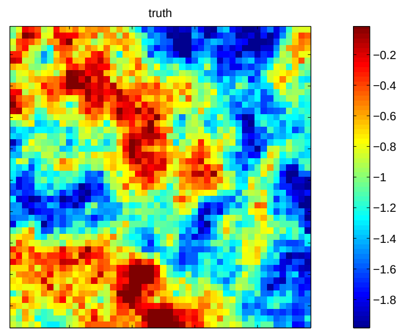

Figure 16: The truth in the practical example

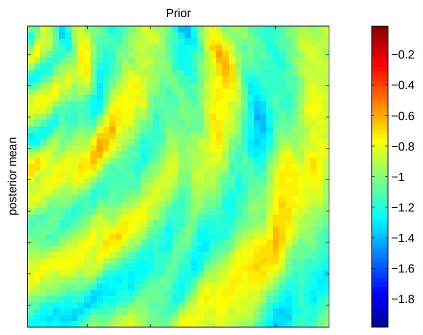

Figure 17: The mean of the ensemble for the practical example

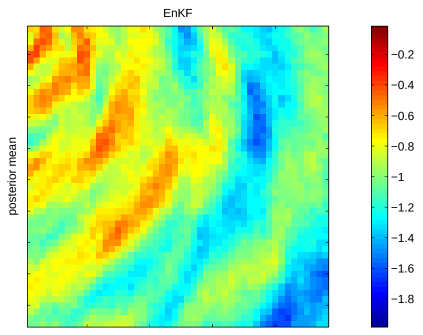

Figure 18: The result of EnKF for the practical example

with the details of the code like numbering the pixels and tree topology. To be consistent, all obtained results are shown as a log10 of permeability values in [meters per day] unit.

4.1 The role of numbering and tree structure

The size of 48×48 ($48 = 3 \times 2^4$) encourages to focus more on the tree construction. This is due to the characteristics of the tree since we want to place equal number of pixels on each finest-scale node and have the same number of children on the nodes in one scale. There were two possibilities tested. One puts 9 pixels per each finest-scale node of a quad-tree ('9 pixels'); the other places 16 pixels per finest-scale node, the root has nine children and all other nodes have four ('9 children'). Figures 19 and 20 show a simple piece of a scheme of both construction methods.

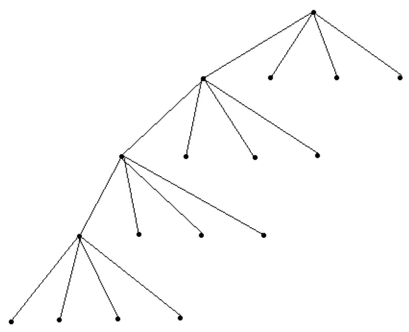
Figure 19: A scheme of a part of a tree structure '9 pixels'

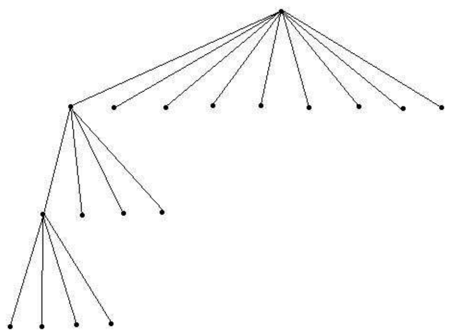
Figure 20: A scheme of a part of a tree structure '9 children'

Various ways of numbering the grid has been tested. Namely, rowwise ('row'), columnwise ('col'), diagonal numbering starting from the main diagonal ('diag') or starting from the other diagonal ('oppd'). One additional way which takes into account the topology of the tree was added to each of the two proposed algorithms ('sqr'): for '9 pixels' and '9 children' a square template of size 9×9 or 16×16, respectively, is ordered on the grid. Since there can be many ideas (probably quite equivalent) to do that, it will not be described in unnecessary details.

Figures from 21 to 30 picture several different results of the EnMSF. In all cases the algorithm is run with the full neighborhood radius. The left column plots are obtained when the '9 children' tree structure is applied, the right column when '9 pixels'. Each pair of pictures in one row is generated with the same numbering type: rowwise, columnwise, diagonal, opposite diagonal and square-like. Some conclusion are given after introducing the following numerical data.

The tables below show the results of assimilation for EnMSF. They are organized similarly as before. The first column indicates the types of numbering used in the assimilation which are described above. Next, there is data of EnMSF: tree identification time, the time of update and the root-mean-square error. The EnKF root-mean-square error is 0.45404 and the error between the truth and the prior is 0.4953. The tables are divided with respect to the tree structure, and the size of the neighborhood: full one and extreme one equal to 1.

Pictures 21-30 might suggest that the second column of results, for '9 pixels', shows more accurate results since they look more smooth and realistic. On the contrary, the RMS error is in general worse for this type of a tree. This might be due to the fact that none of the simulations (except the one with square-like numbering) catches the features in the lower left corner.

Numb.	Tree id.	EnMSF time	RMSE EnMSF
'row'	5.5313	0.29688	0.49621
'col'	5.4375	0.35938	0.49932
'diag'	5.6406	0.25	0.52149
'oppd'	5.6406	0.26563	0.55402
'sqr'	5.7656	0.3125	0.47938

Table 9: Tree structure: '9 children'; Neighborhood: full

Numb.	Tree id.	EnMSF time	RMSE EnMSF
'row'	0.76563	0.32813	0.49794
'col'	0.70313	0.34375	0.49805
'diag'	0.76563	0.32813	0.51629
'oppd'	0.75	0.32813	0.5466
'sqr'	0.78125	0.40625	0.47813

Table 10: Tree structure: '9 children'; Neighborhood: 1

It might be caused by a small number of pixels on the finest scale of the tree where the measurements are placed. The algorithm might not be able to catch dependence on a sufficiently large area. It is different for assimilation with square-like numbering since it keeps the points, which are nearby (dependent) in the reality, close on the dependence tree. This observation might help while designing the numbering scheme for the model where there is some preliminary knowledge of the field.

For the first four rows of plots (21-28) it is visible how the left column with '9 children' tree structure preserves the dependence 'around' the measurements with respect to the numbering scheme.

The tree identification time also depends on the tree structure since all

Numb.	Tree id.	EnMSF time	RMSE EnMSF
'row'	11.0469	0.48438	0.57599
'col'	11	0.5	0.60899
'diag'	11.1719	0.48438	0.56244
'oppd'	11.375	0.625	0.65588
'sqr'	11.2656	0.60938	0.46337

Table 11: Tree structure: '9 pixels'; Neighborhood: full

Numb.	Tree id.	EnMSF time	RMSE EnMSF
'row'	1.0781	0.64063	0.5405
'col'	1.0938	0.65625	0.5219
'diag'	1.0781	0.60938	0.53467
'oppd'	1.0781	0.625	0.6056
'sqr'	1.1875	0.76563	0.45755

Table 12: Tree structure: '9 pixels'; Neighborhood: 1

other parameters are similar. The '9 pixels' method has almost twice as many nodes in the tree (341) as the '9 children' method (190). Then more complex method takes more time. The same rule can be seen for the update time.

Two of the tables above contain the results of the algorithm for a minimal neighborhood equal to 1. Certainly, the time of computing tree parameters decreases significantly. Theoretically though, RMSE results should be worse than the ones with a full neighborhood radius. Paradoxically, the best result so far is obtained for this minimal neighborhood with square-like numbering for '9 pixels' tree (Figure 31). It might suggest that new methods for measuring filter's performance should be found.

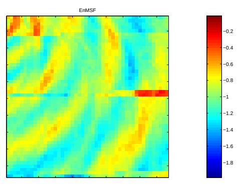
Figure 21: Practical example. The EnMSF estimation obtained for: the tree structure - '9 children'; full neighborhood radius; rowwise numbering

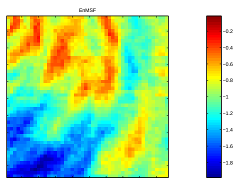
Figure 22: Practical example. The EnMSF estimation obtained for: the tree structure - '9 pixels'; full neighborhood radius; rowwise numbering

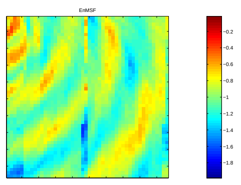
Figure 23: Practical example. The EnMSF estimation obtained for: the tree structure - '9 children'; full neighborhood radius; columnwise numbering

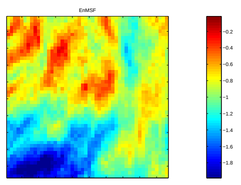
Figure 24: Practical example. The EnMSF estimation obtained for: the tree structure - '9 pixels'; full neighborhood radius; columnwise numbering

For '9 children' structure the RMSE results for two different neighborhood radiuses are almost the same. This might be due to the fact that 16 pixels on each finest scale node keep the most important correlations and a 'smaller' tree can preserve them better. That would be an explanation of why the '9 pixels' tree structure performs better for minimal neighborhood. Since there are so many finest scale nodes (256, where in the other one - 146) with only 9 pixels, when Markov property is weakened, more dependencies can be captured. Some of correlation plots and measures in the next subsection

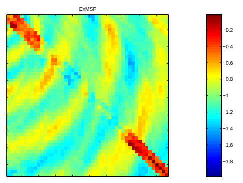

Figure 25: Practical example. The EnMSF estimation obtained for: the tree structure - '9 children'; full neighborhood radius; diagonal numbering

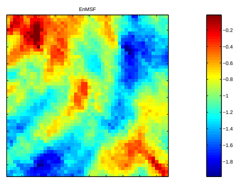

Figure 26: Practical example. The EnMSF estimation obtained for: the tree structure - '9 pixels'; full neighborhood radius; diagonal numbering

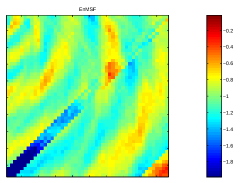

Figure 27: Practical example. The EnMSF estimation obtained for: the tree structure - '9 children'; full neighborhood radius; opposite diagonal numbering

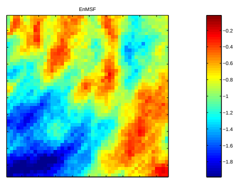

Figure 28: Practical example. The EnMSF estimation obtained for: the tree structure - '9 pixels'; full neighborhood radius; opposite diagonal numbering

might help to judge the results.

Moreover, this Section gives a feeling on how important the tree-construction part is. More research and new ideas should be investigated to build a guide to most efficient EnMSF usage.

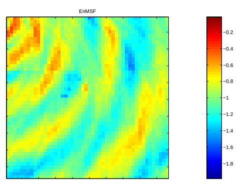

Figure 29: Practical example. The EnMSF estimation obtained for: the tree structure - '9 children'; full neighborhood radius; square-like numbering

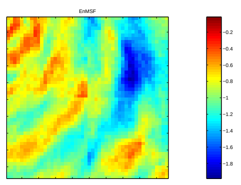

Figure 30: Practical example. The EnMSF estimation obtained for: the tree structure - '9 pixels'; full neighborhood radius; square-like numbering

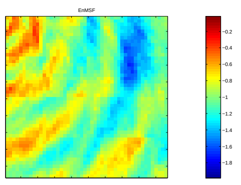

Figure 31: The best result of the practical problem assimilation with the neighborhood 1, square-like numbering, '9 pixels' tree structure

4.2 Correlation

This subsection is going to show a different way of assessing the performance of the multiscale filter which is done on the practical example. Product-moment correlations are computed for each pair of cells x and y:

$$\rho(x,y) = \frac{Cov[x,y]}{\sigma_x \sigma_y},$$

where σ_x is a standard deviation in pixel x. There are three correlation matrices available:

Initial The correlation of the initial ensemble.

After update The correlation of the updated ensemble (EnMSF).

Tree correlation The correlation that EnMSF represents ([1]).

The first two correlations are the standard ones that can be obtained for any filter. The first one is the correlation of the ensemble before update. The second one is the correlation of the ensemble when it is already updated with the measurements. The last one is typical for ensemble multiscale filter and describes how well the tree in the filter represents the initial correlation. To compute it, first, 94 samples need to be drawn on the root of the tree from normal distribution with zero mean and covariance assign to that node. The number of samples is the same as in the initial ensemble. Given the tree parameters for downward recursion equation, the ensemble is propagated to the finest scale and its correlation matrix is computed.

The four tables show comparison results for four different runs of the algorithm with respect to the tree construction and the neighborhood radius. First column is the type of the numbering, second is the comparison between initial and updated correlation, third one is the comparison between initial and tree correlation. The numbers are the percentages of the significantly different values in two matrices. The values are considered different when they have opposite signs and the difference between them is higher or equal to 0.1.

The plots of the absolute value of the correlations, because of technical reasons, show only the last 600×600 square of the matrix. For clarity of the plot only elements with correlation higher than 0.5 are depicted since it makes them easier to compare, and shows the strongest dependencies.

The percentage of difference between initial and updated correlations (second column) assesses the performance of the filter; between initial and tree correlations (third column) shows how well the tree can represent the true dependencies.

Numb.	init.-upd.	init.-tree
'row'	4.4285	15.2684
'col'	3.9417	16.4382
'diag'	3.8208	15.4148
'oppd'	5.0071	15.3390
'sqr'	6.1413	15.0425

Table 13: Correlation[Tree structure: '9 children'; Neighborhood: full]

Numb.	init.-upd.	init.-tree
'row'	3.9536	20.7060
'col'	3.7693	20.7222
'diag'	3.6467	19.7564
'oppd'	5.0126	20.1389
'sqr'	5.4146	24.4686

Table 14: Correlation[Tree structure: '9 children'; Neighborhood: 1]

At last the influence of the neighborhood reduction is visible. For '9 children' tree construction, obviously, it does not influence the predictions. It seems that, although the tree correlation for neighborhood 1 gives a worse representation of the covariance, the tree structure is strong enough to reproduce the correct dependencies, as it was expected from the RMSE in previous section.

For '9 pixels' tree structure the performance of the filter really improves for a smaller neighborhood even though the representation of the correlation

Numb.	init.-upd.	init.-tree
'row'	17.0343	15.5387
'col'	18.1600	13.0535
'diag'	17.7480	17.2672
'oppd'	17.6013	15.0094
'sqr'	17.3072	16.8881

Table 15: Correlation[Tree structure: '9 pixels'; Neighborhood: full]

Numb.	init.-upd.	init.-tree
'row'	14.3689	19.3521
'col'	12.2644	22.5556
'diag'	12.4927	23.6793
'oppd'	13.0133	21.5787
'sqr'	12.8247	22.4282

Table 16: Correlation[Tree structure: '9 pixels'; Neighborhood: 1]

is worse. Evidently the results are still worse than for '9 children'. They confirm some of the RMSE results except for the best one obtained. Before the square-like numbering for '9 pixels' and neighborhood 1 was the most correct.

Just for an example Figure 32 shows the three correlation matrices: initial on the top left, updated on the top right and tree correlation on the bottom.

The updated correlation captures the initial structure very well although given tree matrix has lost some dependencies.

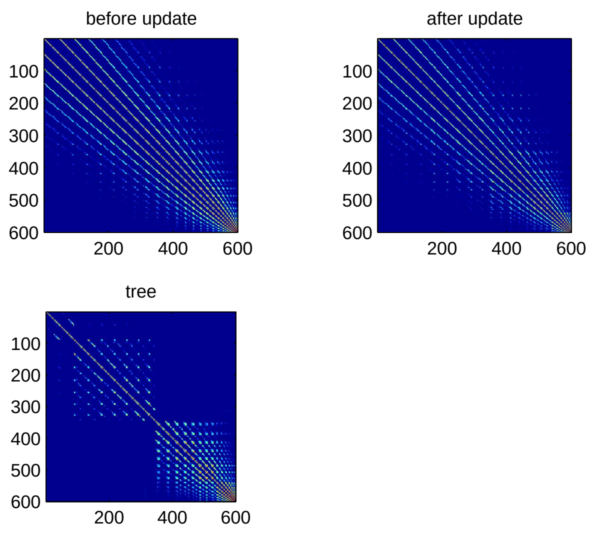

Figure 32: The best result of assimilation for '9 children', diagonal numbering and neighborhood 1

5 Conclusions

The two examples included in the thesis give a better insight into the ensemble multiscale filter's properties. Presented results help to give directions to any further research. Probably, more efficient would be to put more than 9 cells per finest-scale node but the upper bound is not yet known. Pixels should be numbered with respect to a dependence on the field. Then even vague prior knowledge of the field's fluid flow can be included; close numbers could be cumulated in the areas of high dependence or strong flow. Moreover, the filter does not depend very strongly on the number of ensemble members.

It occurred very easy and practical to use whole field measurement. Although very noisy and heavy for EnKF, EnMSF does a good job with those measurements. It is fast and informative since it does not have to work with the full covariance matrix but only its representation. Therefore, the update is done on a set of matrices with low dimension. When the channel is observed the assimilation seems to be more sensitive on the data with respect to the ensemble size. On the other hand, the observation without a channel is almost equally informative for different ensemble size. The possibility of placing the measurement of the mean on the higher scales gave a better result than the finest scale. The possibility of placing any large scale measurements higher in the tree allows to use more than one type of available measurements in one update step.

More information on the measurements' usefulness will also be obtained when the filter is run together with the model simulator. Then the root-mean-square error should be a better indicator of the performance than it is now for one time update step only. That was the reason to additionally look into the correlation matrices what gave a better understanding and more knowledge of the algorithm. Since EnMSF only represents the true covariance matrix, it is important to look into the dependence structure (correlations) as it has been done for the second example.

Definitely, more research needs to be done to design a guide to proper tree construction (the choice of number of pixels or children per node, the best numbering). It will be beneficial to run the algorithm with a model simulator or allow the nodes to overlap. Additionally, a 3D version could be written.

A Appendix

$$A(s) = \widehat{Cov}[\chi(s), \chi(s\gamma)]\widehat{Cov}^{-1}[\chi(s\gamma)]$$

$$F(s) = \widehat{Cov}[\chi(s\gamma)]A(s)^T\widehat{Cov}^{-1}[\chi(s)]$$

$$Q(s) = \widehat{Cov}[\chi(s)] - A(s)\widehat{Cov}[\chi(s), \chi(s\gamma)]^T$$

$$Q'(s) = \widehat{Cov}[\chi(s\gamma)] - F(s)A(s)\widehat{Cov}[\chi(s\gamma)]$$

B Appendix

$$K(s) = \widehat{Cov}[\chi(s), \hat{Y}(s)][\widehat{Cov}[\hat{Y}(s)] + R(s)]^{-1}$$

$$\begin{cases} R(s) = r(s), & \text{m(s)=M;} \\ R(s) = diag[K(s\alpha_1)R(s\alpha_1)K^T(s\alpha_1), ..., K(s\alpha_q)R(s\alpha_q)K^T(s\alpha_q)], & \text{m(s)<M.} \end{cases}$$

$$\begin{cases} Y^j(s) = y(s) + e^j(s), & \text{m(s)=M;} \\ Y(s) = \begin{bmatrix} K(s\alpha_1)Y^j(s\alpha_1) \\ \vdots \\ K(s\alpha_q)Y^j(s\alpha_q) \\ y(s) + e^j(s) \end{bmatrix}, & \text{m(s)<M.} \end{cases}$$

$$\begin{cases} \hat{Y}(s) = h(s)\chi_M^j(s), & \text{m(s)=M;} \\ \hat{Y}(s) = \begin{bmatrix} K(s\alpha_1)\hat{Y}^j(s\alpha_1) \\ \vdots \\ K(s\alpha_q)\hat{Y}^j(s\alpha_q) \\ h(s)\chi_M^j(s) \end{bmatrix}, & \text{m(s)<M.} \end{cases}$$

$$J(s) = \widehat{Cov}[\chi(s|s)]F^T(s)\widehat{Cov}^{-1}[\chi(s\gamma|s)]$$

C Appendix

Model and measurement equations:

$$x(t_{k+1}) = M(t_k)x(t_k) + w(t_k),$$

$$y(t_k) = H(t_k)x(t_k) + v(t_k),$$

$w(t_k)$ - zero-mean normally distributed with covariance $Q(t_k)$; $v(t_k)$ - zero-mean normally distributed with covariance $R(t_k)$.

Kalman filter equations:

model

$$x^f(t_k) = M(t_k)x^a(t_{k-1}),$$

covariance propagation

$$P^f(t_k) = M(t_k)P^a(t_{k-1})M(t_k)^T + Q^T(t_{k-1}),$$

Kalman gain

$$K(t_k) = P^f(t_k)H(t_k)^T(R(t_k) + H(t_k)P^f(t_k)H(t_k)^T)^{-1},$$

analysis

$$x^a(t_k) = x^f(t_k) + K(t_k)(y(t_k) - H(t_k)x^f(t_k)),$$

analyzed covariance

$$P^a(t_k) = [I - K(t_k)H(t_k)]P^f(t_k)[I - K(t_k)H(t_k)]^T + K(t_k)R(t_k)K(t_k)^T,$$

given $x^a(t_0)$ (the initial state) and $P(t_0)$ (covariance of the initial state).

D Appendix

Ensemble Kalman filter equations:

model

$$\xi_i^f(t_k) = M(t_k)\xi_i^a(t_{k-1}) + w_i(t_k),$$

mean

$$x^f(t_k) = \frac{1}{N}\sum_{i=1}^{N} \xi_i^f(t_k),$$

error

$$E^f(t_k) = [\xi_1^f(t_k) - x^f(t_k), ..., \xi_N^f(t_k) - x^f(t_k)],$$

covariance

$$P^f(t_k) = \frac{1}{N-1} E^f(t_k) E^f(t_k)^T,$$

Kalman gain

$$K(t_k) = P^f(t_k)H(t_k)^T(R(t_k) + H(t_k)P^f(t_k)H(t_k)^T)^{-1},$$

analyzed ensemble

$$\xi_i^a(t_k) = \xi_i^f(t_k) + K(t_k)(y(t_k) - H(t_k)\xi_i^f(t_k) + v_i(t_k)),$$

given $\xi_i^a(t_0)$, $i = 1, ..., N$ (an ensemble).

References

[1] Yuhua Zhou, Dennis McLaughlin, Dara Entekhabi, Gene-Hua Crystal Ng. *An Ensemble Multiscale Filter for Large Nonlinear Data Assimilation Problems*

[2] Remus Hanea. *Error Subspaces Filtering for Atmospheric Chemistry Data Assimilation Modeling*

[3] Austin B. Frakt, Alan S. Willsky. *Computationally Efficient Stochastic Realization for Internal Multiscale Autoregressive Models*

[4] Geert K. Brouwer, Peter A. Fokker, Frank Wilschut, Wouter Zijl. *A direct inverse model to determine permeability fields from pressure and flow rate measurements*

[5] Remus Hanea. *Error Subspaces Filtering for Atmospheric Chemistry Data Assimilation Modeling*