

Multi-level Parallelization of Ensemble Kalman Filter for Reservoir History Matching

Reza Tavakoli, Gergina Pencheva, Mary F. Wheeler

Center for Subsurface Modeling, The University of Texas at Austin, Austin, TX

{tavakoli, gergina, mfw}@ices.utexas.edu

1. Abstract

The ensemble Kalman filter (EnKF) has been successfully implemented to assimilate data in reservoir history matching problems. In the EnKF method, a suite of reservoir models (set of ensemble members) runs independently forward in time (forecast step), and is continuously updated as new data becomes available (analysis step) [1].

We have developed a two-level parallel EnKF framework in which multiple realizations are spawned off in parallel on several partitions of a parallel machine (cluster) each of which are further sub-divided among different nodes (processors) and communication performed at data assimilation time, between the partitions before proceeding again to next assimilation step. The analysis step requires collecting a state vector from each ensemble member. If this data is collected on a single processor, this poses an additional limitation on the size of the EnKF problem in terms of both memory and computation time. Therefore, we propose an algorithm in which a third level of parallelization is achieved for the analysis step. The main computational gain of parallelization of the analysis step comes from the fact that the matrix-vector multiplications can be parallelized efficiently.

2. EnKF Methodology

The EnKF algorithm is a sequential Monte Carlo formulation of the Kalman filter, in which an ensemble of reservoir models is used to estimate the correlation between predicted data and reservoir variables. The EnKF update equation for augmented state vector (model parameters and state variables) is given for $j = 1, 2, \dots, N_e$,

$$y_j^{n,a} = y_j^{n,f} + C_{Y^n D^n}^f (C_{D^n D^n}^f + C_{D^n}^f)^{-1} (d_{uc,j}^n - d_j^{n,f}).$$

The cross-covariance matrices can be expressed by

$$C_{Y^n D^n}^f = \frac{1}{N_e - 1} \sum_{j=1}^{N_e} (y_j^{n,f} - \bar{y}^{n,f}) (d_j^{n,f} - \bar{d}^{n,f})^T,$$

$$C_{D^n D^n}^f = \frac{1}{N_e - 1} \sum_{j=1}^{N_e} (d_j^{n,f} - \bar{d}^{n,f}) (d_j^{n,f} - \bar{d}^{n,f})^T,$$

where the predicted data $d_j^{n,f}$ is obtained by running the reservoir simulator forward, which is represented by the non-linear equation

$$d_j^{n,f} = g_n(m_j).$$

3. Parallel EnKF

3.1 Parallelization of Forecast Step

In a serial implementation of EnKF, N_e simulation runs are done sequentially as shown in Fig 1. Running N_e simulations in sequence greatly amplifies the wall-clock time between data assimilation steps. This motivates the need of

using parallelization in the EnKF algorithm [2, 3]. Since the stepping forward of the ensemble members are independent, the forecast step can be parallelized by running an ensemble of reservoir models concurrently on a different processor (or node) of a parallel computer. Moreover, implementation of a parallel reservoir simulator (IPARS)[4] further improves the speed of the process (Fig. 1).

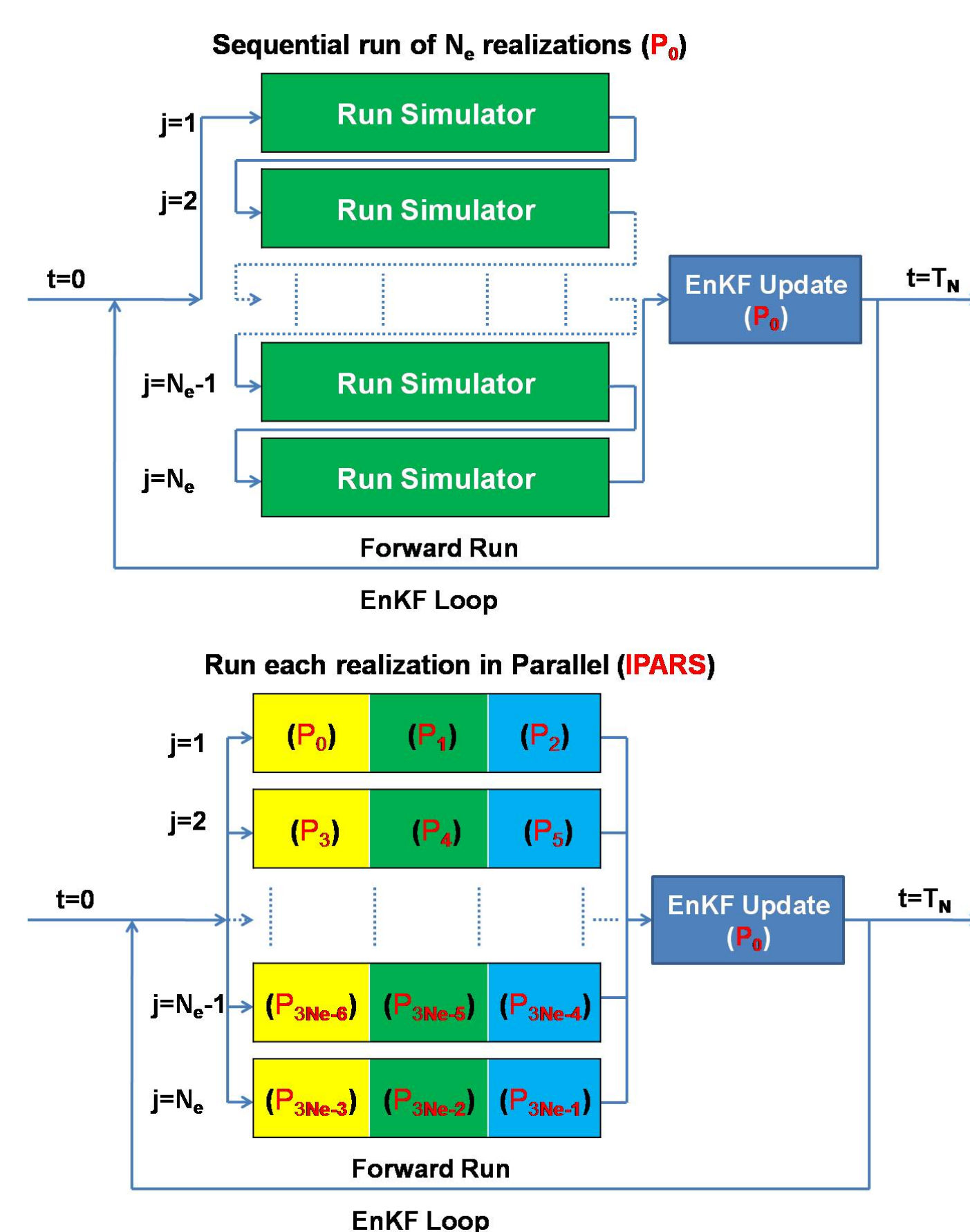


Figure 1: Flow diagrams of serial EnKF (top) versus parallel EnKF (bottom)

3.2 Parallelization of Update Step

Rewrite the EnKF update equation in terms of the matrix of ensemble members $Y^n = [y_1^n \ y_2^n \ \dots \ y_{N_e}^n]$, yields [5]

$$Y^{n,a} = Y^{n,f} \{ I + (\Delta D^n)^T [\Delta D^n (\Delta D^n)^T + (N_e - 1) C_{D^n}^f]^{-1} (D_{uc}^n - D^n) \}$$

$$= Y^{n,f} \{ I + \delta(D) \},$$

where $\Delta D^n = D^n - \bar{D}^n$, $D^n = [d_1^n \ d_2^n \ \dots \ d_{N_e}^n]$, and

$$\delta(D) = (\Delta D^n)^T [\Delta D^n (\Delta D^n)^T + (N_e - 1) C_{D^n}^f]^{-1} (D_{uc}^n - D^n).$$

Note:

- all rows of $Y^{n,a}$ are updated with the same coefficient matrix
- the updating coefficient matrix is only a function of data ($N_d \ll N_y$)
- $\delta(D)$ is an $N_e \times N_e$ with $N_e \ll N_y$

Therefore, the parallelization of update step is summarized as follows:

- at each assimilation time, a predicted data vector from each realization must be communicated to one specific processor to construct matrix D^n

- this processor builds D_{uc}^n by reading observation data and then computes the updating coefficient matrix $(I + \delta(D))$
- the updating coefficient matrix is communicated to all central processors (with ids $0, 1, \dots, p - 1$)
- Y^n is partitioned row-wise into p pieces and each partition is assigned to one of the central processors
- the update step is performed in parallel on the central processors as shown below

$$\begin{aligned} \text{Node 0} &\Rightarrow \begin{bmatrix} Y^0 \\ Y^1 \\ \vdots \\ Y^{p-1} \end{bmatrix}^{n,a} = \begin{bmatrix} Y^0 \\ Y^1 \\ \vdots \\ Y^{p-1} \end{bmatrix}^{n,f} (I + \delta(D)) \\ \text{Node 1} &\Rightarrow \\ \vdots & \\ \text{Node } p-1 &\Rightarrow \end{aligned}$$

- the updated state vectors are sent back to the processors which are assigned to perform the next forecast step

4. Computational Results

4.1 Example 1

- Two-phase, 2D reservoir
- $20 \times 30 = 600$ grid blocks
- 5 producers and 2 injectors
- Simulation time is 2700 days with update interval of 30 days
- Observation data are bottom hole pressure (BHP), oil production rate and water-oil ratio (WOR)
- Horizontal permeability and porosity as model parameters
- Used 100 ensemble members

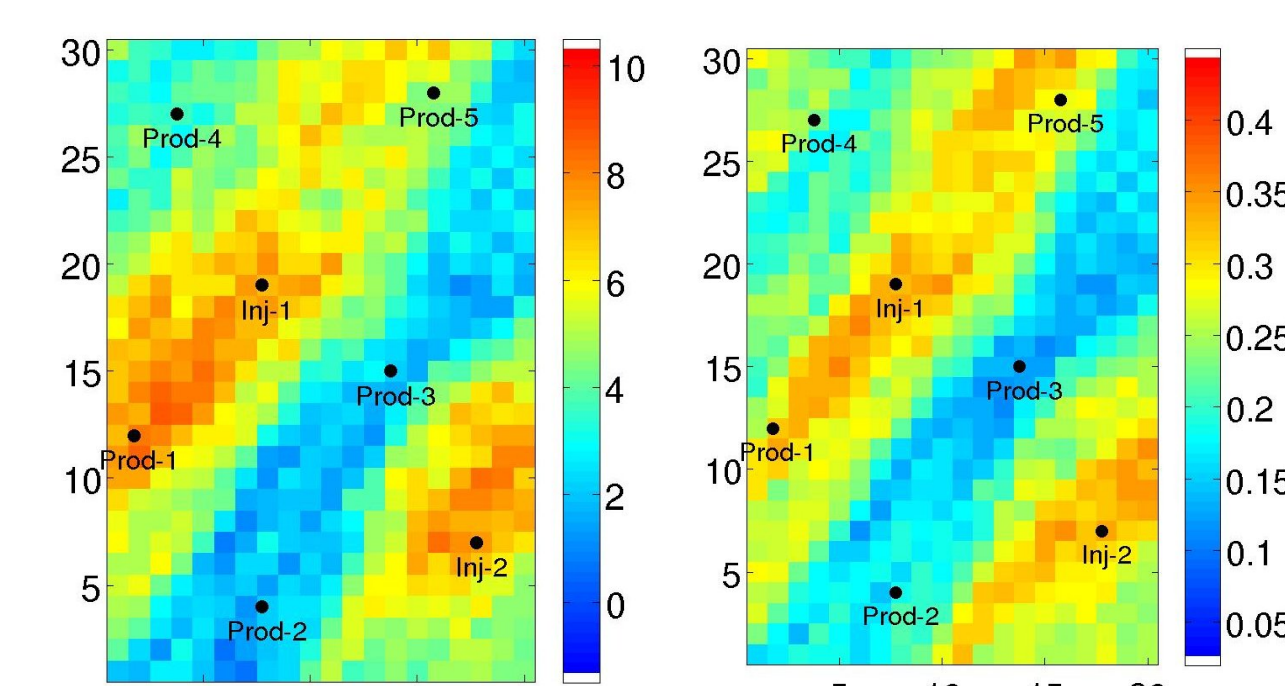


Figure 2: True permeability and true porosity

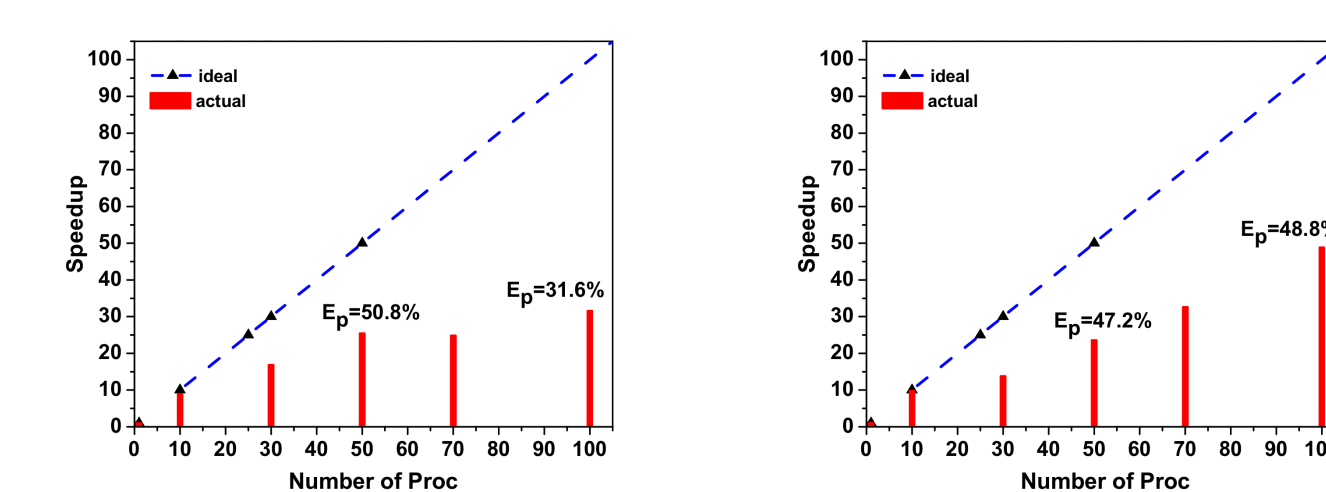


Figure 3: Speedup and parallel efficiency on Lonestar (left) and Ranger (right) clusters

Clusters	Number of processors					
	1	10	30	50	70	100
Lonestar	10.4	1.18	0.62	0.41	0.42	0.33
Ranger	13.7	1.40	0.99	0.58	0.42	0.28

Table 1: Execution time(hrs) as a function of number of processors

4.2 Example 2

- Two-phase, 3D reservoir
- $50 \times 50 \times 4 = 10,000$ grid blocks
- 7 producers and 2 injectors
- Simulation time is 4500 days with update interval of 100 days
- Used different ensemble sizes

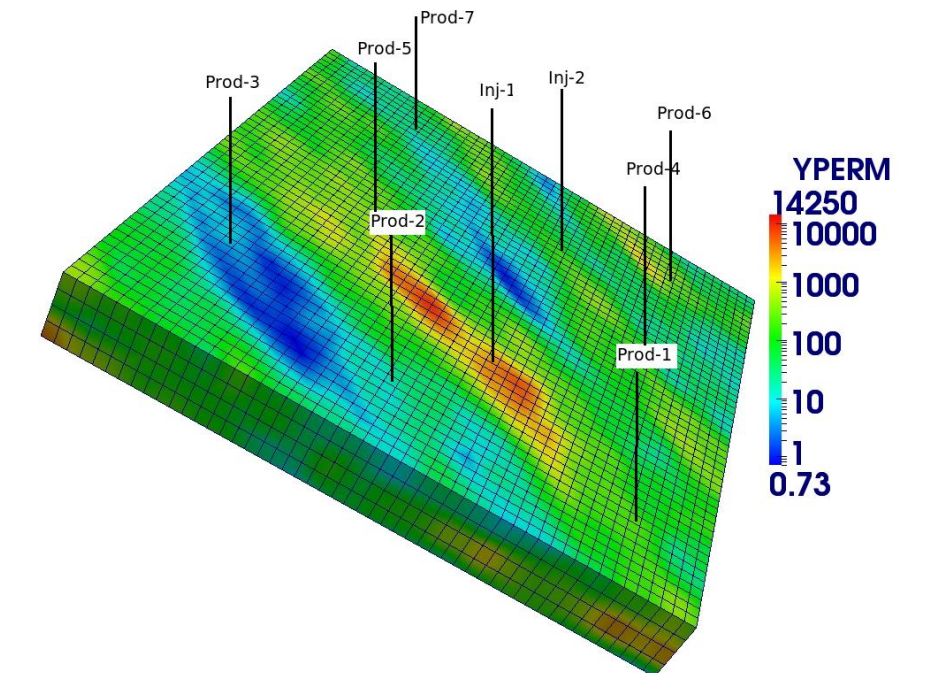


Figure 4: True horizontal permeability field with the well locations

Serial Run	Ne=50		
	Npsim=1	Npsim=2	Npsim=3
14.77	1.28	0.97	0.78

Table 2: Execution time(hrs) on Bevo2

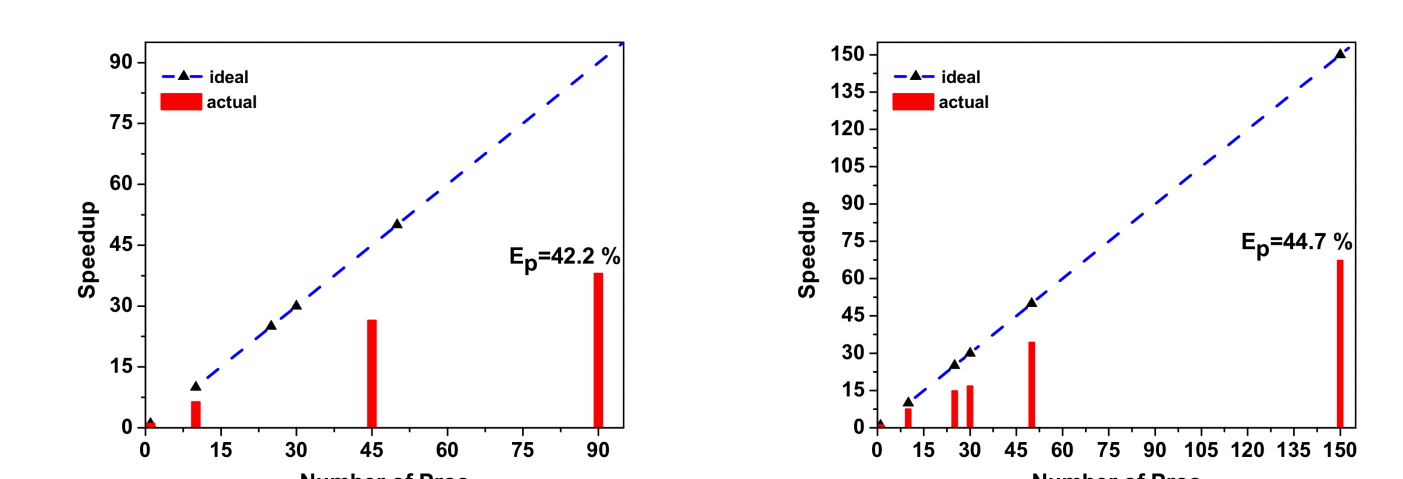


Figure 5: Speedup and parallel efficiency on Bevo2 with $N_e = 90$ (left) and $N_e = 150$ (right)

Clusters	Ne=90				Ne=150			
	1	10	45	90	1	10	50	150
Bevo2	33	5.2	1.25	0.87	39.9	5.26	1.14	0.57
Ranger	>12	18.5	5.17	2.27	>12	16.7	4.66	2.12
Lonestar	>12	13.43	2.71	1.64	>12	13.92	3.75	1.75

Table 3: Execution time(hrs) as a function of number of processors

5. Conclusions

An efficient parallel EnKF framework has been developed which uses two-level parallelization of the forecast step. Moreover, we have presented an algorithm for parallelization of the update step. We have obtained more than 40% parallel efficiency by parallelization of the forecast step. As the size of the problem increases, more meaningful timing is obtained.

References

- [1] G. Evensen. *Data Assimilation: The Ensemble Kalman Filter*. Springer, 2007.
- [2] P. L. Houtekamer and H. L. Mitchell. A sequential ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 129(1):123–137, 2001.
- [3] C. L. Keppenne. Data assimilation into a primitive-equation model with a parallel ensemble Kalman filter. *Monthly Weather Review*, 128(6):1971–1981, 2000.
- [4] J. Wheeler et al. *IPARS User's Manual*. Center for Subsurface Modeling, ICES, The University of Texas at Austin, Dec 10 2002.
- [5] M. Zafari and A. C. Reynolds. Assessing the uncertainty in reservoir description and performance predictions with the ensemble Kalman filter. *SPE Journal*, 12(3):382–391, 2007. SPE 95750-PA.