NTNU | Norwegian University of Science and Technology

A SPARSE MATRIX FORMULATION OF THE MODEL-BASED ENSEMBLE KALMAN FILTER

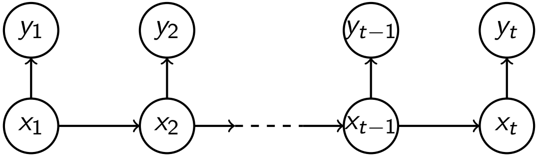
Håkon Gryvill

Joint work with Håkon Tjelmeland

Outline

1. Standard EnKF
2. Issues with standard EnKF
3. Model-based EnKF
4. Computational issues with model-based EnKF
5. New strategy
6. Results
7. Closing remarks

Introduction - State space model

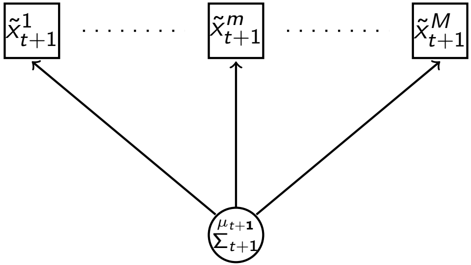


- Variables of interest $x_1, \ldots, x_t \in \mathbb{R}^n$. High dimensional.
- $x_1 \sim p(x_1)$
- Forward model $x_{t+1} = g(x_t, \epsilon_t)$
- Observations $y_1, \ldots, y_t \in \mathbb{R}^m$
- Observation model $y_t | x_t \sim N(Hx_t, R)$
- Aim: *filtering* distribution $p(x_t | y_1, \ldots, y_t)$

Introduction - Update step

Part 1: Compute $\tilde{\mu}_{t+1}$ and $\tilde{\Sigma}_{t+1}$

- Assume $\tilde{x}^1_{t+1}, \ldots, \tilde{x}^M_{t+1} | \mu_{t+1}, \Sigma_{t+1} \overset{iid}{\sim} N(\mu_{t+1}, \Sigma_{t+1})$
- Approximate μ_{t+1} and Σ_{t+1} by $\tilde{\mu}_{t+1}$ and $\tilde{\Sigma}_{t+1}$



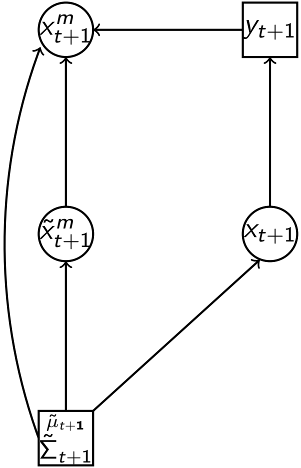
Introduction - Update step

Part 2: Update $\tilde{x}^m_{t+1}$ into x^m_{t+1}

- Assume $\tilde{x}^m_{t+1}, x_{t+1} | \tilde{\mu}_{t+1}, \tilde{\Sigma}_{t+1} \overset{iid}{\sim} N(\tilde{\mu}_{t+1}, \tilde{\Sigma}_{t+1})$
- $y_{t+1} | x_{t+1} \sim N(Hx_{t+1}, R)$
- Require $x^m_{t+1} | \tilde{\mu}_{t+1}, \tilde{\Sigma}_{t+1}, y_{t+1} \overset{d}{=} x_{t+1} | \tilde{\mu}_{t+1}, \tilde{\Sigma}_{t+1}, y_{t+1}$

Satisfied by standard EnKF update:

$$x^m_{t+1} = \tilde{x}^m_{t+1} + \tilde{K}_{t+1}(y_{t+1} + \tilde{\epsilon}^m_{t+1} - H\tilde{x}^m_{t+1})$$



Introduction - Update step

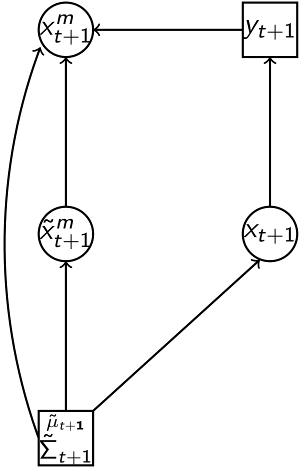
Part 2: Update $\tilde{x}^m_{t+1}$ into x^m_{t+1}

- Assume $\tilde{x}^m_{t+1}, x_{t+1} | \tilde{\mu}_{t+1}, \tilde{\Sigma}_{t+1} \overset{iid}{\sim} N(\tilde{\mu}_{t+1}, \tilde{\Sigma}_{t+1})$
- $y_{t+1} | x_{t+1} \sim N(Hx_{t+1}, R)$
- Require

  $x^m_{t+1} | \tilde{\mu}_{t+1}, \tilde{\Sigma}_{t+1}, y_{t+1} \overset{d}{=} x_{t+1} | \tilde{\mu}_{t+1}, \tilde{\Sigma}_{t+1}, y_{t+1}$

Also satisfied by square root filter:

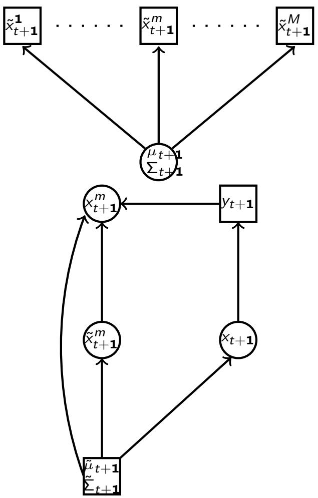
$$x^m_{t+1} = \tilde{\mu}_{t+1} + \tilde{K}_{t+1}(y_{t+1} - H\tilde{\mu}_{t+1}) + B(\tilde{x}^m_{t+1} - \tilde{\mu}_{t+1}),$$
$$B\tilde{\Sigma}_{t+1}B^T = (I - \tilde{K}_{t+1}H)\tilde{\Sigma}_{t+1}$$



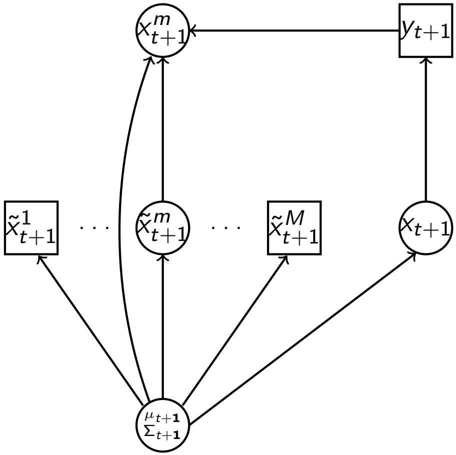
Introduction - Issues

- Uncertainty in $\tilde{\mu}_{t+1}, \tilde{\Sigma}_{t+1}$ is ignored
  - Solution proposed in Myrseth and Omre (2010) and Tsyrulnikov and Raktiko (2017)
- Information in $\tilde{x}_{t+1}^{m}$ is used twice (inconsistently)
  - Solution proposed in Houtekamer and Mitchell (1997)
- Information in y_{t+1} about μ_{t+1}, Σ_{t+1} is ignored
  - Discussed in Myrseth and Omre (2010), but ignored
- Why this EnKF update?
- EnKF tends to underestimate uncertainty

Model-based EnKF - Model-based update of $\tilde{x}^m_{t+1}$ to x^m_{t+1}



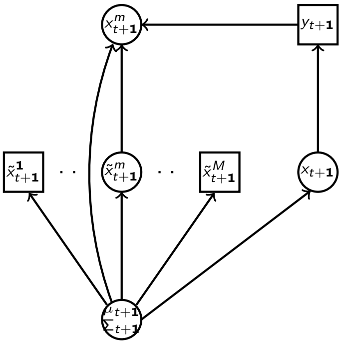
$\Longrightarrow$



Introduced in Loe and Tjelmeland (2020)

Model-based EnKF - Model-based update of $\tilde{x}^m_{t+1}$ to x^m_{t+1}

$$
\begin{aligned}
\tilde{x}^1_{t+1}, \ldots, \tilde{x}^M_{t+1} | \mu_{t+1}, \Sigma_{t+1} &\overset{iid}{\sim} N(\mu_{t+1}, \Sigma_{t+1}) \\
(\mu_{t+1}, \Sigma_{t+1}) &\sim \text{NIW}(\mu_0, \lambda, \Psi, \nu) \\
y_{t+1} | x_{t+1} &\sim N(Hx_{t+1}, R)
\end{aligned}
$$



Require:

$$
x^m_{t+1} | \tilde{x}^1_{t+1}, \ldots, \tilde{x}^{m-1}_{t+1}, \tilde{x}^{m+1}_{t+1}, \ldots, \tilde{x}^M_{t+1}, y_{t+1} \overset{d}{=} x_{t+1} | \tilde{x}^1_{t+1}, \ldots, \tilde{x}^{m-1}_{t+1}, \tilde{x}^{m+1}_{t+1}, \ldots, \tilde{x}^M_{t+1}, y_{t+1}
$$

Optimality criterion: Minimise

$$
\text{E}\left[(x^m_{t+1} - \tilde{x}^m_{t+1})^T (x^m_{t+1} - \tilde{x}^m_{t+1}) \right]
$$

Model-based EnKF - Model-based update of $\tilde{x}_{t+1}^m$ to x_{t+1}^m

Algorithm

- Sample $\mu_{t+1}^m, \Sigma_{t+1}^m | \tilde{x}_{t+1}^1, \ldots, \tilde{x}_{t+1}^{m-1}, \tilde{x}_{t+1}^{m+1}, \ldots, \tilde{x}_{t+1}^M, y_{t+1} \sim \mathrm{NIW}(\mu_0^\star, \lambda^\star, \Psi^\star, \nu^\star)$
- Compute Kalman gain K_{t+1}^m
- Compute weight matrix B_{t+1}^m
- Update

$$x_{t+1}^m = \mu_{t+1}^m + B_{t+1}^m(\tilde{x}_{t+1}^m - \mu_{t+1}^m) + K_{t+1}^m(y_{t+1} - H\mu_{t+1}^m)$$

Model-based EnKF - Model-based update of $\tilde{x}_{t+1}^m$ to x_{t+1}^m

Algorithm

- Sample $\mu_{t+1}^m, \Sigma_{t+1}^m | \tilde{x}_{t+1}^1, \dots, \tilde{x}_{t+1}^{m-1}, \tilde{x}_{t+1}^{m+1}, \dots, \tilde{x}_{t+1}^M, y_{t+1} \sim \text{NIW}(\mu_0^\star, \lambda^\star, \Psi^\star, \nu^\star)$
- Compute Kalman gain K_{t+1}^m
- Compute weight matrix B_{t+1}^m
- Update

$$x_{t+1}^m = \mu_{t+1}^m + B_{t+1}^m(\tilde{x}_{t+1}^m - \mu_{t+1}^m) + K_{t+1}^m(y_{t+1} - H\mu_{t+1}^m)$$

Results

- Provides reliable results with realistic uncertainty representation
- However: Computationally demanding

Computational issues - Prior for model parameters

Recall:

- $(\mu_{t+1}, \Sigma_{t+1}) \sim \text{NIW}(\mu_0, \lambda, \Psi, \nu)$
- $\tilde{x}_{t+1}^1, \ldots, \tilde{x}_{t+1}^M | \mu_{t+1}, \Sigma_{t+1} \overset{\text{iid}}{\sim} N(\mu_{t+1}, \Sigma_{t+1})$

$\Longrightarrow \; \mu_{t+1}^m, \Sigma_{t+1}^m | \tilde{x}_{t+1}^1, \ldots, \tilde{x}_{t+1}^{m-1}, \tilde{x}_{t+1}^{m+1}, \ldots, \tilde{x}_{t+1}^M, y_{t+1} \sim \text{NIW}(\mu_0^\star, \lambda^\star, \Psi^\star, \nu^\star)$

Computational issues - Prior for model parameters

Recall:

- $(\mu_{t+1}, \Sigma_{t+1}) \sim \text{NIW}(\mu_0, \lambda, \Psi, \nu)$
- $\tilde{x}_{t+1}^1, \ldots, \tilde{x}_{t+1}^M | \mu_{t+1}, \Sigma_{t+1} \overset{\text{iid}}{\sim} N(\mu_{t+1}, \Sigma_{t+1})$

$$\implies \mu_{t+1}^m, \Sigma_{t+1}^m | \tilde{x}_{t+1}^1, \ldots, \tilde{x}_{t+1}^{m-1}, \tilde{x}_{t+1}^{m+1}, \ldots, \tilde{x}_{t+1}^M, y_{t+1} \sim \text{NIW}(\mu_0^\star, \lambda^\star, \Psi^\star, \nu^\star)$$

Issues:

- Sampling from $\text{NIW}(\mu_0^\star, \lambda^\star, \Psi^\star, \nu^\star)$ is computationally demanding
- Σ_{t+1}^m is a full matrix

Computational issues - Prior for model parameters

Recall:

- $(\mu_{t+1}, \Sigma_{t+1}) \sim \text{NIW}(\mu_0, \lambda, \Psi, \nu)$
- $\tilde{x}_{t+1}^1, \ldots, \tilde{x}_{t+1}^M | \mu_{t+1}, \Sigma_{t+1} \overset{\text{iid}}{\sim} N(\mu_{t+1}, \Sigma_{t+1})$

$\Longrightarrow \mu_{t+1}^m, \Sigma_{t+1}^m | \tilde{x}_{t+1}^1, \ldots, \tilde{x}_{t+1}^{m-1}, \tilde{x}_{t+1}^{m+1}, \ldots, \tilde{x}_{t+1}^M, y_{t+1} \sim \text{NIW}(\mu_0^\star, \lambda^\star, \Psi^\star, \nu^\star)$

Issues:

- Sampling from $\text{NIW}(\mu_0^\star, \lambda^\star, \Psi^\star, \nu^\star)$ is computationally demanding
- Σ_{t+1}^m is a full matrix

Solution:

- Use sparse precision matrix $Q_{t+1} = \Sigma_{t+1}^{-1}$
- Choose distribution such that
  1. $Q_{t+1} | \tilde{x}_{t+1}^1, \ldots, \tilde{x}_{t+1}^{m-1}, \tilde{x}_{t+1}^{m+1}, \ldots, \tilde{x}_{t+1}^M, y_{t+1}$ can be sampled efficiently
  2. $Q_{t+1} | \tilde{x}_{t+1}^1, \ldots, \tilde{x}_{t+1}^{m-1}, \tilde{x}_{t+1}^{m+1}, \ldots, \tilde{x}_{t+1}^M, y_{t+1}$ becomes sparse

Computational issues - Computing weight matrix B^m_{t+1}

Recall:

$$x^m_{t+1} = \mu^m_{t+1} + B^m_{t+1}(\tilde{x}^m_{t+1} - \mu^m_{t+1}) + K^m_{t+1}(y_{t+1} - H\mu^m_{t+1})$$

"Optimal update"

Computational issues - Computing weight matrix B^m_{t+1}

Recall:

$$x^m_{t+1} = \mu^m_{t+1} + B^m_{t+1}(\tilde{x}^m_{t+1} - \mu^m_{t+1}) + K^m_{t+1}(y_{t+1} - H\mu^m_{t+1})$$

"Optimal update"

We compute B^m_{t+1} as follows

1. Cholesky decomposition $VV^T = Q_{t+1}$
2. Cholesky decomposition $UU^T = Q_{t+1} + H^T RH$
3. Compute $Z = V^T U$
4. Compute singular value decomposition $Z = PGF^T$
5. Compute $B^m_{t+1} = U^{-T} FP^T V^T$

Computational issues - Computing weight matrix B_{t+1}^m

Recall:

$$x_{t+1}^m = \mu_{t+1}^m + B_{t+1}^m(\tilde{x}_{t+1}^m - \mu_{t+1}^m) + K_{t+1}^m(y_{t+1} - H\mu_{t+1}^m)$$

"Optimal update"

We compute B_{t+1}^m as follows

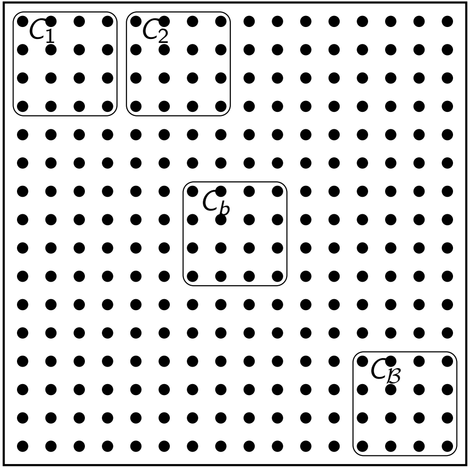
1. Cholesky decomposition $VV^T = Q_{t+1}$
2. Cholesky decomposition $UU^T = Q_{t+1} + H^T RH$
3. Compute $Z = V^T U$
4. Compute singular value decomposition $Z = PGF^T$
5. Compute $B_{t+1}^m = U^{-T} FP^T V^T$

Issue:

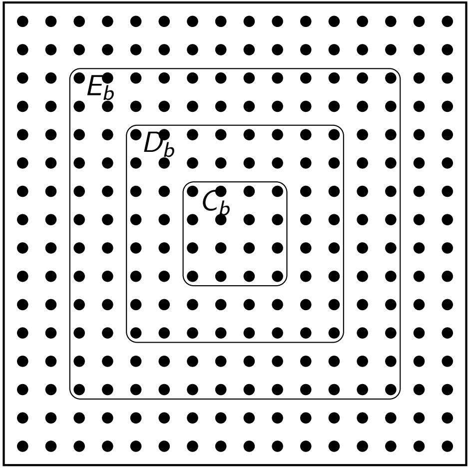
Computing step 4 is computationally demanding when Z is large

Computational issues - Block update

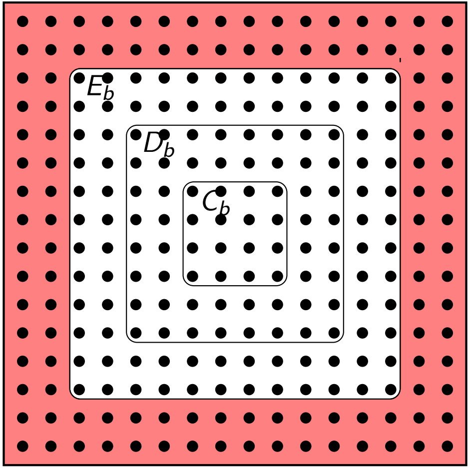
Resembles domain localisation, but the motivation is different



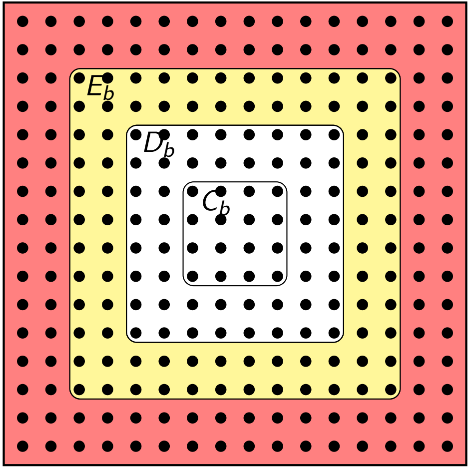
Computational issues - Block update



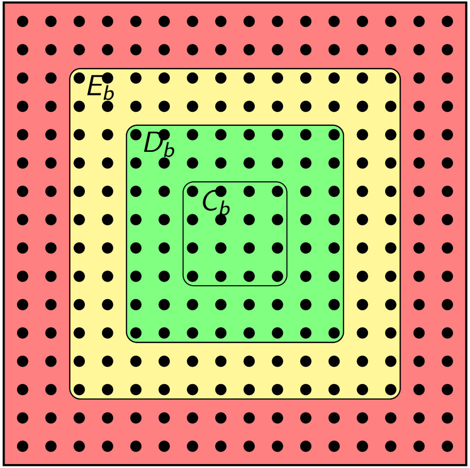
Computational issues - Block update



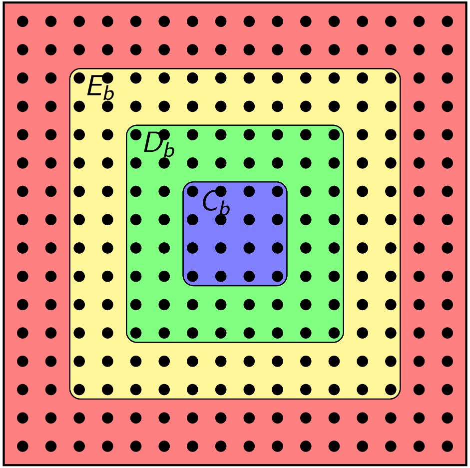
Computational issues - Block update



Computational issues - Block update



Computational issues - Block update



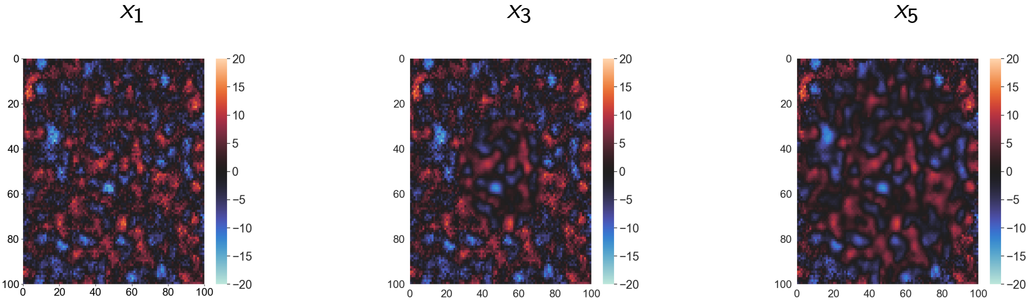
Simulation examples - Aim

Aim:
Compare the optimal update and block update

- Computational demands
- Results

Simulation examples - Setup

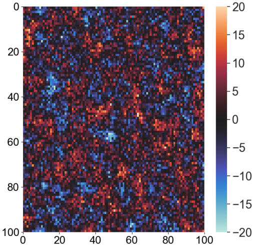
- State vector x_t constructed in a grid of size $s \times s$
- x_1 is Gaussian field, spatial correlation structure
- Forward function: Deterministic, smoothing around center node
- Vague prior for μ and Q. Same for all time steps
- Observations: local average, additive Gaussian noise



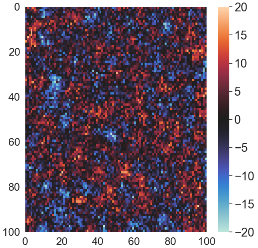
Simulation examples - Setup

- State vector x_t constructed in a grid of size $s \times s$
- x_1 is Gaussian field, spatial correlation structure
- Forward function: Deterministic, smoothing around center node
- Vague prior for μ and Q. Same for all time steps
- Observations: local average, additive Gaussian noise

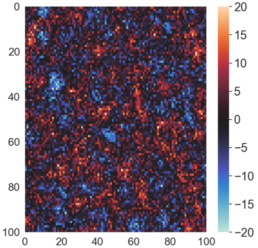
y_1



y_3

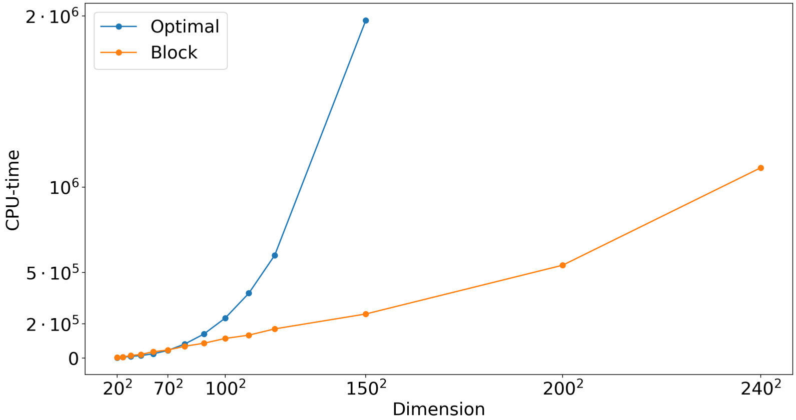


y_5



Simulation examples - Comparison of CPU-times

- Run both update procedures on grids of sizes $20 \times 20, 30 \times 30, \ldots, 120 \times 120$ and 150×150.
- Additionally, run block update with $200 \times 200, 240 \times 240$



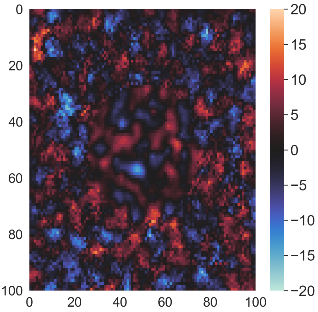
Simulation examples - Stepwise comparison

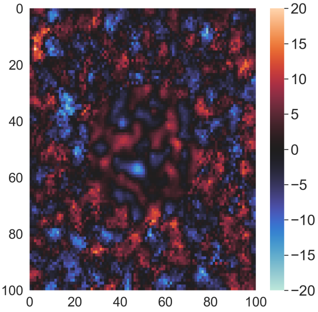
- Run both update procedures on a 100×100-grid
- Block update: blocks of size 20×20
- The two ensembles are updated using the same forecast ensemble, observations and model parameters

Optimal

$t = 3$

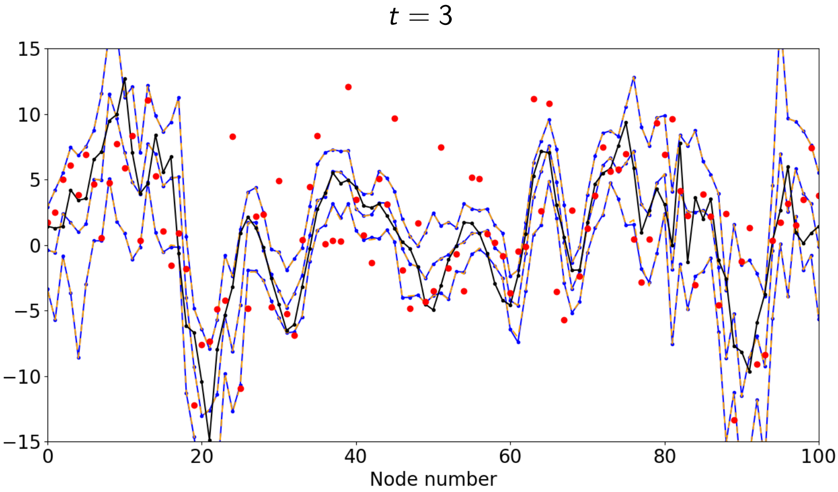
Block





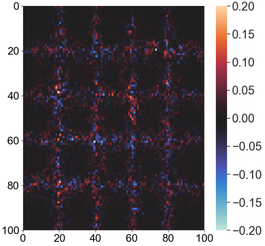
Simulation examples - Stepwise comparison

- Comparison along one cross section of the grid

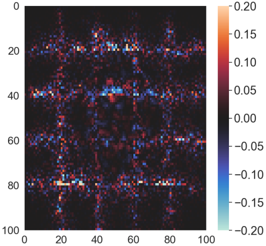


Simulation examples - Stepwise comparison

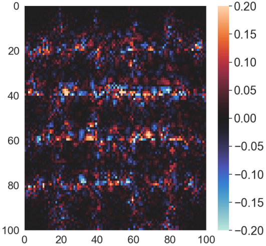
$t = 1$



$t = 3$



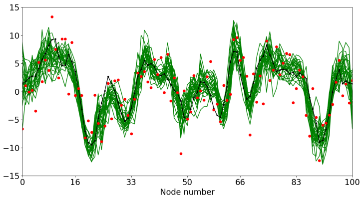
$t = 5$



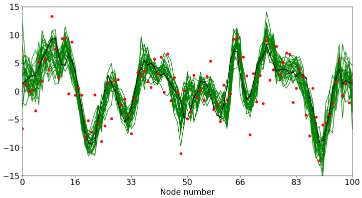
Simulation examples - Comparing different simulations

- Run both update procedures for $t = 5$ iterations

Optimal



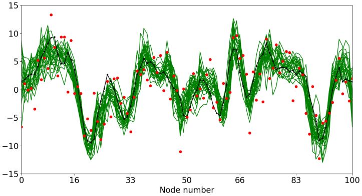
Block

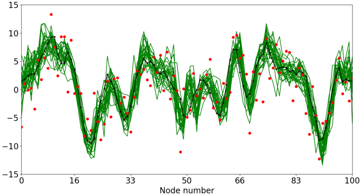


Simulation examples - Comparing different simulations

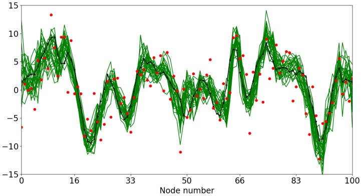
- Run both update procedures for $t = 5$ iterations

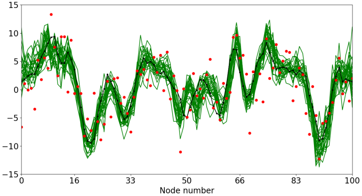
Optimal





Block

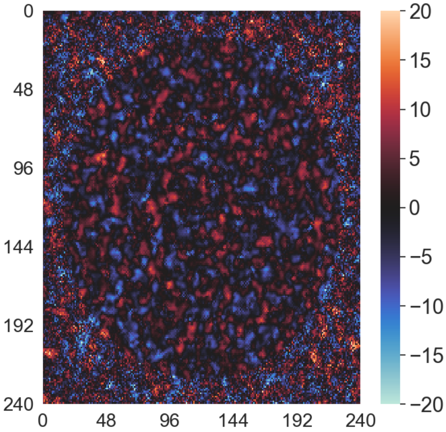




Simulation examples - High-dimensional simulation example

- Run block update on a grid of size 240×240 for $t = 10$ iterations

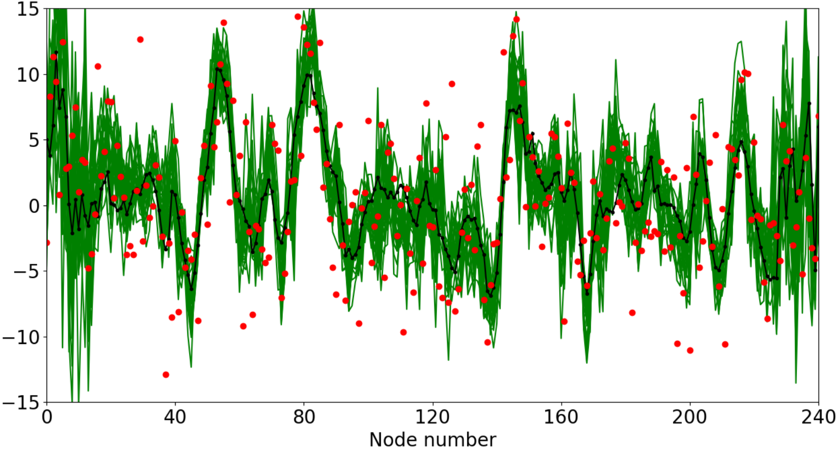
$$t = 9$$



Simulation examples - High-dimensional simulation

- One cross section

$$t = 9$$



Closing remarks - Summary

New strategy

- Formulate model using precision matrices
- Sparse precision matrices
- Block update

Results

- Block update faster than optimal
- Block update provides essentially similar results