PDAF Parallel Data Assimilation Framework

ALFRED-WEGENER-INSTITUT
HELMHOLTZ-ZENTRUM FÜR POLAR- UND MEERESFORSCHUNG

Ensemble-Based Data Assimilation: Concepts, Methods, and Hands-On Tutorials

Lecture 3: Advanced Ensemble Kalman Filters

Lars Nerger

Alfred Wegener Institute
Helmholtz Center for Polar and Marine Research
Bremerhaven, Germany

HELMHOLTZ
SPITZENFORSCHUNG FÜR GROSSE HERAUSFORDERUNGEN

MarDATA Workshop on Data Assimilation, Bremerhaven, July 24-25, 2024

Overview

Overall aim of the workshop: Get familiar with concepts of data assimilation and selected methods for high-dimensional models

Yesterday

1. Data Assimilation Basics
2. Ensemble Data Assimilation

Today

3. Advanced Ensemble Kalman filters
    - hands-on tutorial
        - break at ~10:30h
    - hands-on tutorial
4. Practical aspects

Overview – Lecture 3

Discuss advanced filter variants suitable for large-scale data assimilation + 'fixes' needed to make them usable

- Ensemble transform Kalman filter - ETKF
- Inflation
- Localization

Ensemble Transform Kalman Filter

(one of the current efficient ‘work horses’)

Square-roots

Forecast ensemble covariance matrix:

$$\mathbf{P}_k^f = \frac{1}{N-1}\mathbf{X}_k^{f\prime}(\mathbf{X}_k^{f\prime})^T$$

Analysis ensemble covariance matrix:

$$\mathbf{P}_k^a = \frac{1}{N-1}\mathbf{X}_k^{f\prime}\mathbf{A}(\mathbf{X}_k^{f\prime})^T$$

What is $\mathbf{A}$?

with symmetric matrix $\mathbf{A}$

Analysis ensemble perturbations then given by

$$\mathbf{X}_k^{a\prime} = \mathbf{X}_k^{f\prime}\mathbf{A}^{1/2}$$

Square-root is computed by eigenvalue decomposition (EVD)

$$\mathbf{A} = \mathbf{V}\mathbf{S}\mathbf{V}^T \rightarrow \mathbf{A}^{1/2} = \mathbf{V}\mathbf{S}^{1/2}\mathbf{V}^T$$

← symmetric square root

V: eigenvectors; **S**: diagonal matrix of eigenvalues

Ensemble transformations (2)

Possibilities to obtain $\mathbf{X}_k^a$

1. **Monte Carlo analysis update**
   - Kalman update of each single ensemble member

2. **Explicit ensemble transformation**
   1. Kalman update of ensemble mean state
   2. Transformation of ensemble perturbations $\mathbf{X}' = \mathbf{X} - \bar{\mathbf{X}}$
      a. Right sided: $\mathbf{X}'^a = \mathbf{X}'^f \mathbf{W}$ small matrix $\mathbf{W}$ (size $N \times N$)
      b. Left sided: $\mathbf{X}'^a = \hat{\mathbf{W}} \mathbf{X}'^f$ large matrix $\hat{\mathbf{W}}$ (size $n \times n$) (needs tricks to be usable)

Right sided ensemble transformation

$$\mathbf{X}'^{a} = \mathbf{X}'^{f}\mathbf{W}$$

Very common - used in:

- SEIK (Singular Evolutive Interpolated KF, Pham et al. 1998)
- ETKF (Ensemble Transform KF, Bishop et al. 2001)
- EnsRF (Ensemble Square-root Filter, Whitaker/Hamill 2001)
- ESTKF (Error-subspace Transform KF, Nerger et al 2012)

Very efficient: $\mathbf{W}$ is small ($N \times N$)

Efficient use of ensembles

The original EnKF is simple, but not efficient because of this:

$\mathbf{P}_k^f$ can be approximated by ensemble or modes: $\tilde{\mathbf{P}}_k^f$

Analysis at time t_k:

$$\mathbf{x}_k^a = \mathbf{x}_k^f + \tilde{\mathbf{K}}_k \left(\mathbf{y}_k - \mathbf{H}_k \mathbf{x}_k^f \right)$$

Kalman gain

$$\tilde{\mathbf{K}}_k = \tilde{\mathbf{P}}_k^f \mathbf{H}_k^T \left(\mathbf{H}_k \tilde{\mathbf{P}}_k^f \mathbf{H}_k^T + \mathbf{R}_k \right)^{-1}$$

Costly inversion: $m \times m$ matrix!

➜ Degrees of freedom given by ensemble size, not number of observations!

➜ Ensembles allow for cost reduction – if **R** is invertible at low cost

Efficient use of ensembles

$\mathbf{P}_k^f$ can be approximated by ensemble or modes: $\tilde{\mathbf{P}}_k^f = (N-1)^{-1}\mathbf{X}'\mathbf{X}'^T$

Analysis at time t_k:

$$\mathbf{x}_k^a = \mathbf{x}_k^f + \tilde{\mathbf{K}}_k \left(\mathbf{y}_k - \mathbf{H}_k \mathbf{x}_k^f \right)$$

Kalman gain

$$\tilde{\mathbf{K}}_k = \tilde{\mathbf{P}}_k^f \mathbf{H}_k^T \underbrace{\left(\mathbf{H}_k \tilde{\mathbf{P}}_k^f \mathbf{H}_k^T + \mathbf{R}_k \right)^{-1}}$$

Costly inversion: $m \times m$ matrix!

Alternative (using Sherman-Morrison-Woodbury identity)

$$\tilde{\mathbf{K}}_k = \mathbf{X}' \underbrace{\left[(N-1)\mathbf{I} + \mathbf{X}'^T \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \mathbf{X}' \right]^{-1}} \mathbf{X}'^T \mathbf{H}^T \mathbf{R}^{-1}$$

Inversion of $N \times N$ matrix

$\mathbf{X}' = \mathbf{X} - \bar{\mathbf{X}}$

From previous lecture: BLUE in Vector Form – Error of State Estimate

Analysis state

$$\mathbf{x}^a = \mathbf{x}^b + (\mathbf{B}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{H}\mathbf{x}^b)$$

Equivalent (using Sherman-Morrison-Woodbury identity)

$$\mathbf{x}^a = \mathbf{x}^b + \mathbf{B}\mathbf{H}^T(\mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{H}\mathbf{x}^b)$$

Now define the *gain matrix* as

$$\mathbf{K} = \mathbf{B}\mathbf{H}^T\left(\mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{R}\right)^{-1} \quad \text{thus} \quad \mathbf{x}^a = \mathbf{x}^b + \mathbf{K}(\mathbf{y} - \mathbf{H}\mathbf{x}^b)$$

$$= \mathbf{P}^a\mathbf{H}^T\mathbf{R}^{-1} \qquad = \mathbf{x}^b + \mathbf{P}^a\mathbf{H}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{H}\mathbf{x}^b)$$

Analysis error

$$\mathbf{P}^a = (\mathbf{I} - \mathbf{K}\mathbf{H})\,\mathbf{B} \quad = \left(\mathbf{B}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}$$

$$= \mathbf{X}'\underbrace{\left[(N-1)\mathbf{I} + \mathbf{X}'^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{X}'\right]}_{\mathbf{A}^{-1}}{}^{-1}\mathbf{X}'^T$$

This alternative form can be useful when we factorize **B = LL^T**

Ensemble Transform Kalman Filter - ETKF

Ensemble perturbation matrix *(computed explicitly)*

$$\mathbf{X}'_k := \mathbf{X}_k - \overline{\mathbf{X}_k}$$

size (n x N)

Analysis covariance matrix *(we use it, but don't compute it)*

$$\mathbf{P}^a = \mathbf{X}'^f \mathbf{A} (\mathbf{X}'^f)^T$$

(n x n)

"Transform matrix" (in ensemble space)

$$\mathbf{A}^{-1} := (N-1)\mathbf{I} + (\mathbf{H}\mathbf{X}'^f)^T \mathbf{R}^{-1} \mathbf{H}\mathbf{X}'^f$$

(N x N)

Ensemble transformation

$$\mathbf{X}'^a = \mathbf{X}'^f \ \mathbf{W}^{ETKF}.$$

(n x N)

Ensemble weight matrix

$$\mathbf{W}^{ETKF} := \sqrt{N-1}\mathbf{A}^{1/2}$$

(N x N)

- Use EVD $\mathbf{A}^{-1} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T \quad \rightarrow \mathbf{A}^{1/2} = \mathbf{U}\mathbf{\Sigma}^{-1/2}\mathbf{U}^T$

ETKF (2) – Updating the ensemble mean

Kalman gain (at time k, $\mathbf{X}' = \mathbf{X}^f - \overline{\mathbf{X}}^f$)

$$\mathbf{K} = \mathbf{X}' \underbrace{\left[(N-1)\,\mathbf{I} + \underbrace{\mathbf{X}'^T \mathbf{H}^T \mathbf{R}^{-1}}_{\mathbf{Z}} \mathbf{H}\mathbf{X}' \right]^{-1}}_{\mathbf{A}^{-1}} \underbrace{\mathbf{X}'^T \mathbf{H}^T \mathbf{R}^{-1}}_{\mathbf{Z}}$$

State (ensemble mean) update

$$\mathbf{x}^a = \mathbf{x}^f + \mathbf{X}' \mathbf{A} \underbrace{\mathbf{X}'^T \mathbf{H}^T \mathbf{R}^{-1}}_{\mathbf{Z}} \left(\mathbf{y} - \mathbf{H}\mathbf{x}^f \right)$$

Need to invert $\mathbf{A}^{-1}$

- reuse eigenvalue decomposition: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{U}^T$

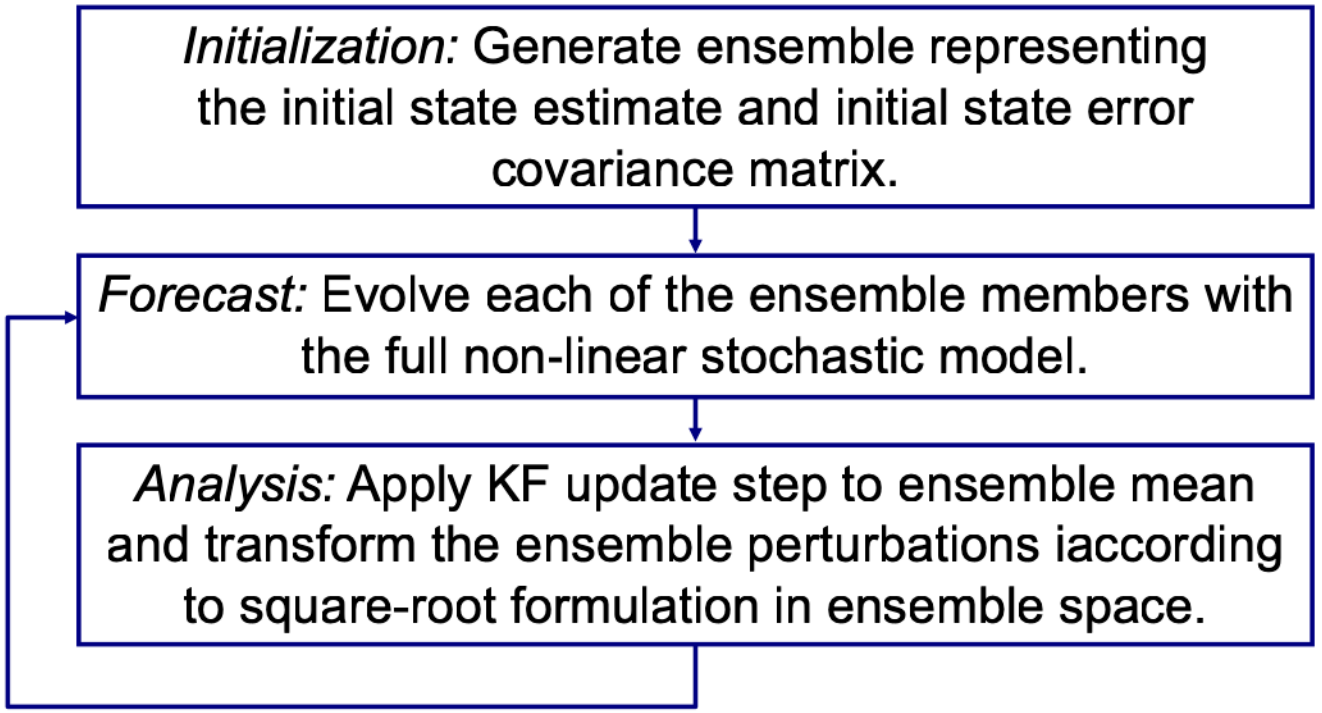
For efficiency: reuse $\mathbf{Z} = \mathbf{X}'^T \mathbf{H}^T \mathbf{R}^{-1}$ already computed for $\mathbf{A}^{-1}$

Full state update

$$\mathbf{x}^a = \mathbf{x}^f + \mathbf{X}'\mathbf{w}$$

$$\mathbf{w} = \mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{U}^T \mathbf{Z} \left(\mathbf{y} - \mathbf{H}\mathbf{x}^f \right)$$

The ETKF



The ETKF with localization (LETKF, Hunt et al. 2007) seems to be the currently most widely used EnKF variant

Properties of the ETKF

- Computational complexity
  - linear in dimension *n* of state vector (Important since *n* is huge!)
  - approx. linear in dimension *m* of observation vector (*m* is still large!)
  - cubic with ensemble size *N* (*N* is usually the smallest dimension)

- Low complexity due to explicit consideration of ensemble spanned space (error subspace):
  - ⇨ Degrees of freedom given by ensemble size–1
  - ⇨ Analysis increment: combination of ensemble members with weight computed in error subspace

- Compared to EnKF
  - no sampling noise from observation ensemble
  - Cheaper inversion of small matrix (*N x N* for ETKF)
  - need to perform eigenvalue decomposition of small matrix

Essential “Fixes”

Covariance Inflation

Localization

Limitations of pure ensemble filters

Only ensemble size O(100) is feasible to use

→ Ensemble size is always much smaller than state dimension

→ We always have an extreme low-rank approximation of $\mathbf{P}$

This leads to

- sampling errors in $\mathbf{P}^f$
- underestimation of variances in $\mathbf{P}^f$
- inefficient analysis step
- regional error increases due to analysis step
- strong underestimation of variances in $\mathbf{P}^a$

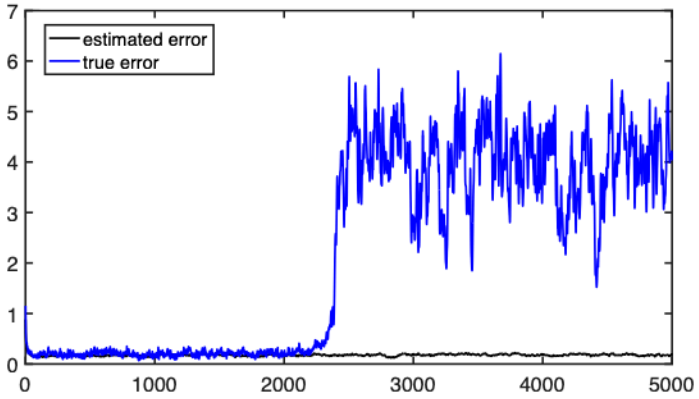
Covariance Inflation

Filter Divergence

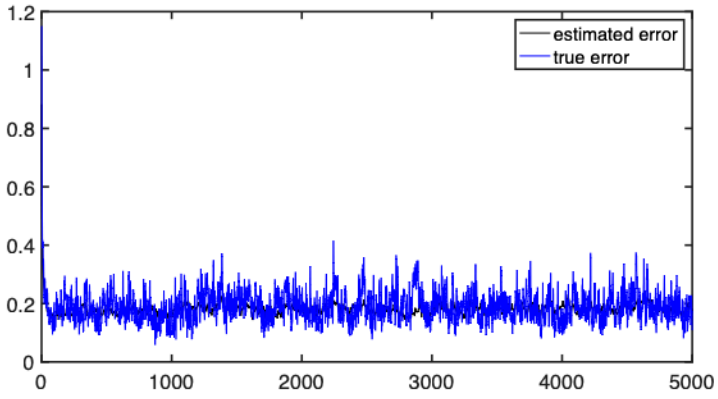
- Ensemble spread estimates the assimilation error
- **Converged filter**: true error and estimate error are similar
- **Filter divergence**: large true error, but small ensemble spread
  - Filter is off-track, but fails to notice this
  - ➜ can be caused by insufficient ensemble variance

Example from Lorenz-96 model

Filter divergence



Converged filter



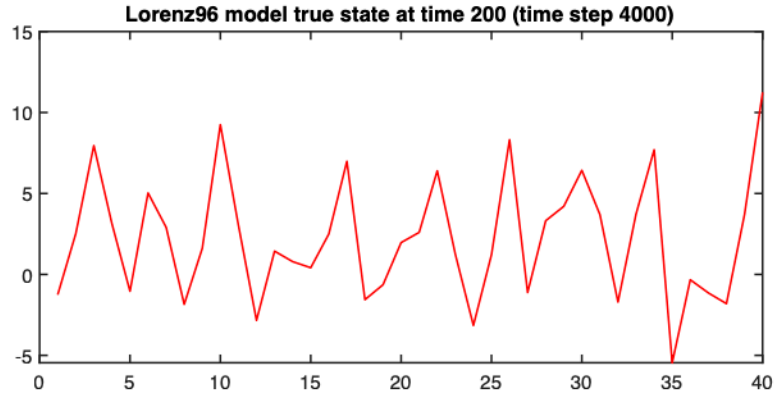
Lorenz-96 Model (Lorenz, 1996)

- non-dimensional set of equations with period boundary condition

$$\frac{dx_j}{dt} = (x_{j+1} - x_{j-2})x_{j-1} - x_j + F$$

$j = 1, \ldots, J$ is the grid point index

- Typical configuration
  - J = 40, F=8 ➜ In this case, the model is chaotic
- Time stepper: Runge-Kutta 4th order, dt=0.05



1-dimensional chaotic wave

Lorenz-96 Model (Lorenz, 1996) (II)

- Dynamics of Lorenz-96 model
- ~8 waves fit into domain
  - Drift westward (toward domain index 0)
  - Intended to represent Rossby waves
  - Detailed behavior is chaotic

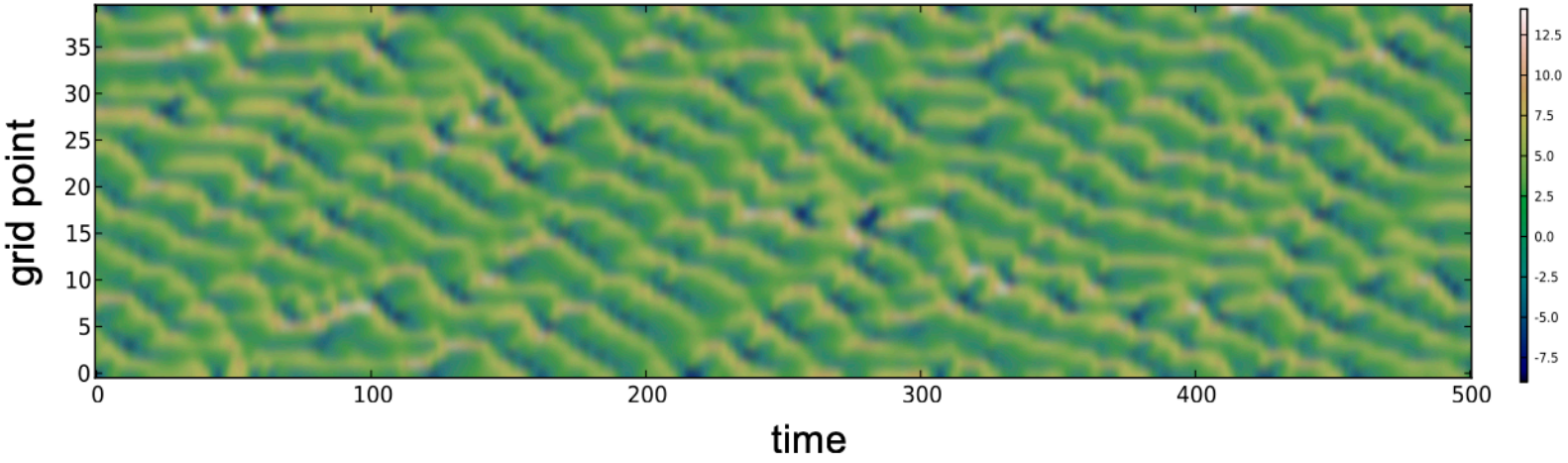


Figure: M. Bocquet, Lecture notes

Excursion: Twin experiments

Real assimilation cases:

- We don't know the true system state
- Observation errors are not exactly known
- Observations and model can have biases (systematic errors)

Idealize for data assimilation – twin experiments:

- Generate synthetic observations from model run
- Use model state and add prescribed error
  (e.g. Gaussian with prescribed variance)
- Then perform data assimilation; choose initial state which deviate from model run used to generate observations

OSSEs – Observation system simulation experiments

- Synthetic observations simulate real ones
- Can systematically deteriorate the system
  (e.g. using different model resolutions, observation bias)

Covariance inflation

- True variance is always underestimated
  - finite ensemble size
  - sampling errors (unknown structure of P)
  - model errors

  → can lead to filter divergence

- Simple remedy

  → Increase error estimate before analysis

- Possibilities
  - Increase ensemble spread (“inflation”)
  - Multiply covariance matrix by a factor slightly above 1
  - Additive error term (e.g. on diagonal)

(Mathematically, this is a regularization)

Multiplicative inflation and forgetting factor

- Multiplicate ensemble inflation

$$\mathbf{X} \rightarrow \mathbf{X} + \alpha(\mathbf{X} - \overline{\mathbf{X}}) \qquad \alpha \geq 1$$

(applied to large $n \times N$ matrix)

- Computationally more efficient in ETKF and ESTKF: *Forgetting factor*

$$\mathbf{A}^{-1} := \rho\,(N-1)\mathbf{I} + (\mathbf{H}\mathbf{X}'^{f})^{T}\mathbf{R}^{-1}\mathbf{H}\mathbf{X}'^{f} \qquad \rho \leq 1$$

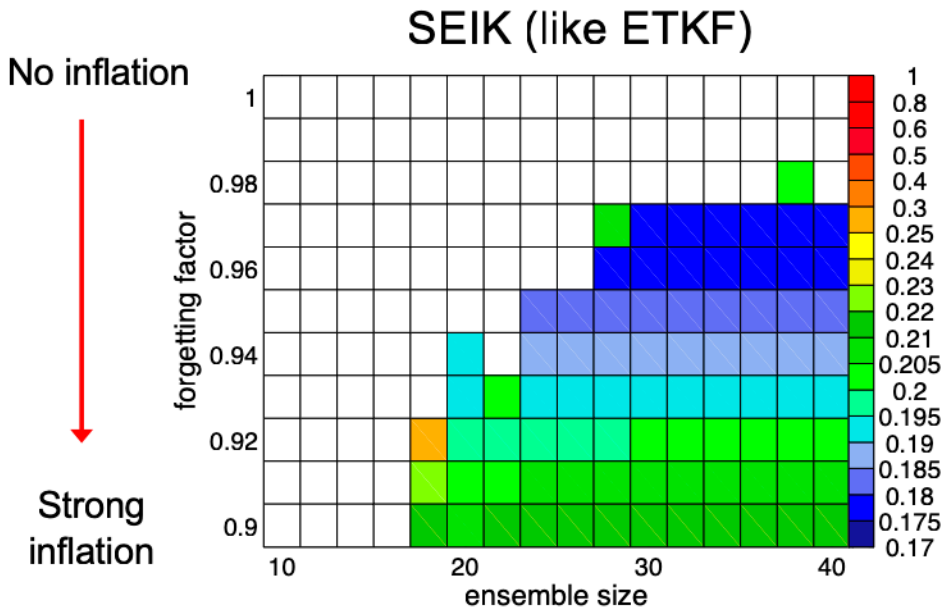
(applied to small $N \times N$ matrix)

Relation: $$\rho = \frac{1}{\alpha^2}$$

- Using ρ inflates with constant factor over full state vector
- Using α might allow applying varying factors

Impact of inflation on stability & performance

Experiments with Lorenz96 model

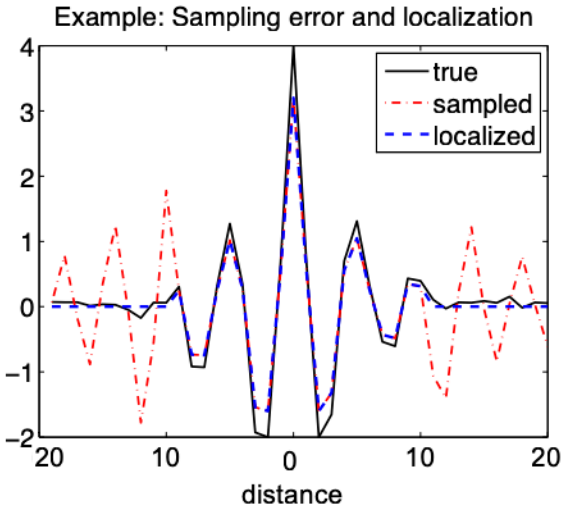


- white: filter fails („diverges“)
- increased stability with stronger inflation (smaller forgetting factor)
- optimal choice for inflation factor

Localization

Localization: Why and how?

- Combination of observations and model state based on estimated error covariance matrices
- Finite ensemble size leads to significant sampling errors
  - particularly for zero or small covariances!



- Remove estimated long-range correlations
  - ➜ Increases degrees of freedom for analysis (globally not locally!)
  - ➜ Increases size of analysis correction

(introduced for EnKFs by Houtekamer & Mitchell 1998)

Localization Types

Analysis equation:

$$\mathbf{x}^a = \mathbf{x}^f + \frac{\mathbf{P}^f\mathbf{H}^T}{\mathbf{H}\mathbf{P}^f\mathbf{H}^T + \mathbf{R}}\left(\mathbf{y} - \mathbf{H}\mathbf{x}^f\right)$$

Covariance localization

- Modify covariances in forecast covariance matrix $\mathbf{P}^f$
- Element-wise product with correlation matrix of compact support

Requires that $\mathbf{P}^f$ is computed (OK in EnKF, but not in ETKF)

Observation localization

- Modify observation error covariance matrix $\mathbf{R}$
- Needs distance of observation (achieved by local analysis or domain localization)

Possible in all filter formulations; common choice in ETKF, ESTKF

Covariance Localization (localize $\mathbf{P}$)

Apply localization to state error covariance matrix $\tilde{\mathbf{P}}$

- Generate localization weight matrix C
  using covariance functions (e.g. Normal function)
- Apply with element-wise product

$$\tilde{\mathbf{P}}_{loc} = \mathbf{C} \circ \tilde{\mathbf{P}}$$

Practical approach without computing $\tilde{\mathbf{P}}$

- Apply $\mathbf{C}$ to observed matrices:

$$(\mathbf{H}\tilde{\mathbf{P}}\mathbf{H}^T)_{loc} = \mathbf{H}\mathbf{C}\mathbf{H}^T \circ \mathbf{H}\tilde{\mathbf{P}}\mathbf{H}^T$$
$$(\tilde{\mathbf{P}}\mathbf{H}^T)_{loc} = \mathbf{C}\mathbf{H}^T \circ \tilde{\mathbf{P}}\mathbf{H}^T$$

Here $\mathbf{H}\mathbf{C}\mathbf{H}^T$ and $\mathbf{C}\mathbf{H}^T$ are usually implemented as operators, not full matrices

Observation Localization

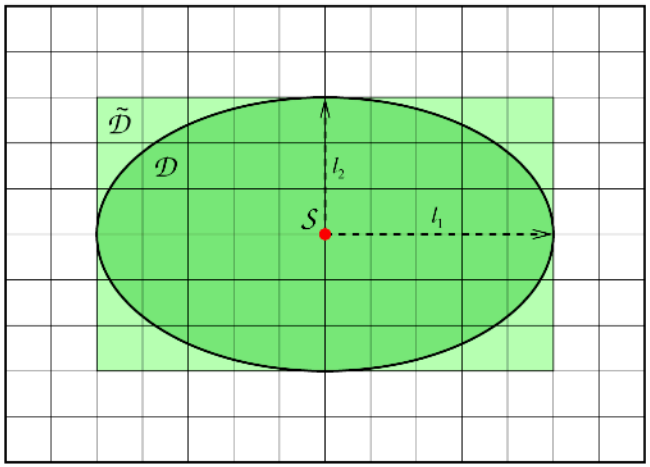
Local Analysis:

- Update small regions
  (like single vertical columns)
  allows to define distance

  ➜ This needs a loop over
  local analysis domains

- Use only observation within some
  distance around this region

- State update and ensemble
  transformation fully local

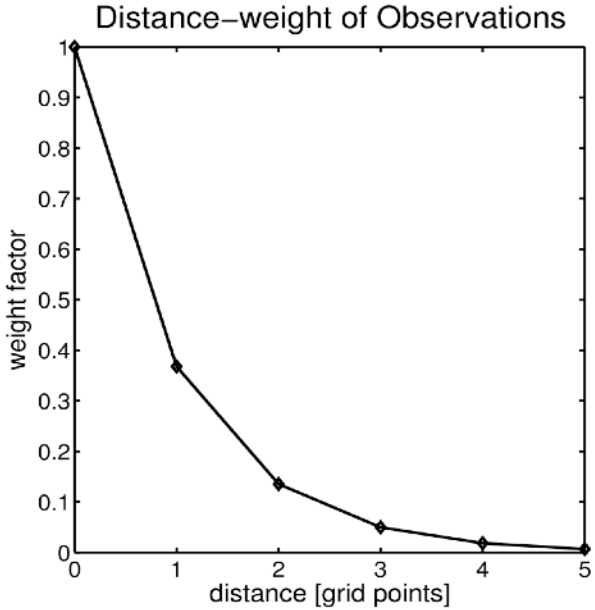


S: Analysis region
D: Corresponding data region

Observation localization

Localizing weight

- use local analysis
  (allows to define distance)
- reduce weight for remote
  observations by increasing
  variance estimates
- use e.g. exponential decrease
  or polynomial representing
  correlation function of compact
  support

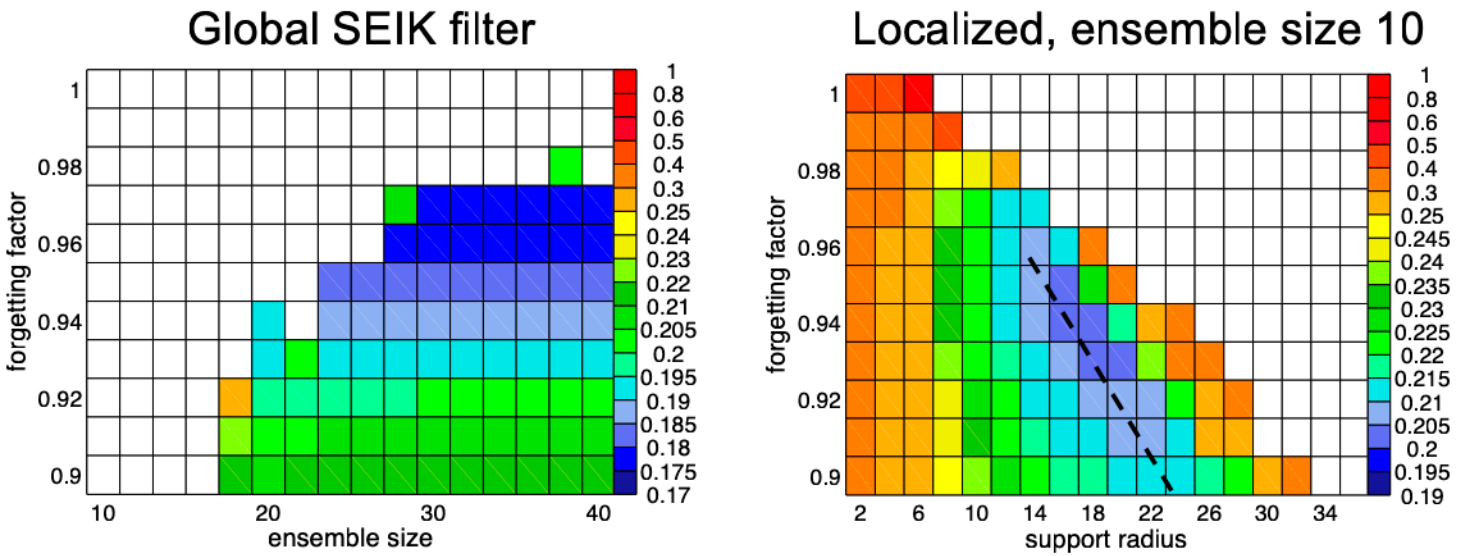


- In ETKF/ESTKF we apply the localization weight as

$$\mathbf{R}_{loc}^{-1} = \hat{\mathbf{C}} \circ \mathbf{R}^{-1}$$

Impact of inflation and localization

Experiments with Lorenz96 model



- smaller ensemble usable with localization
- optimal combination of forgetting factor and support radius exists

Optimal localization radius

- Localization radius is typically chosen by
  - experience
  - testing different radii

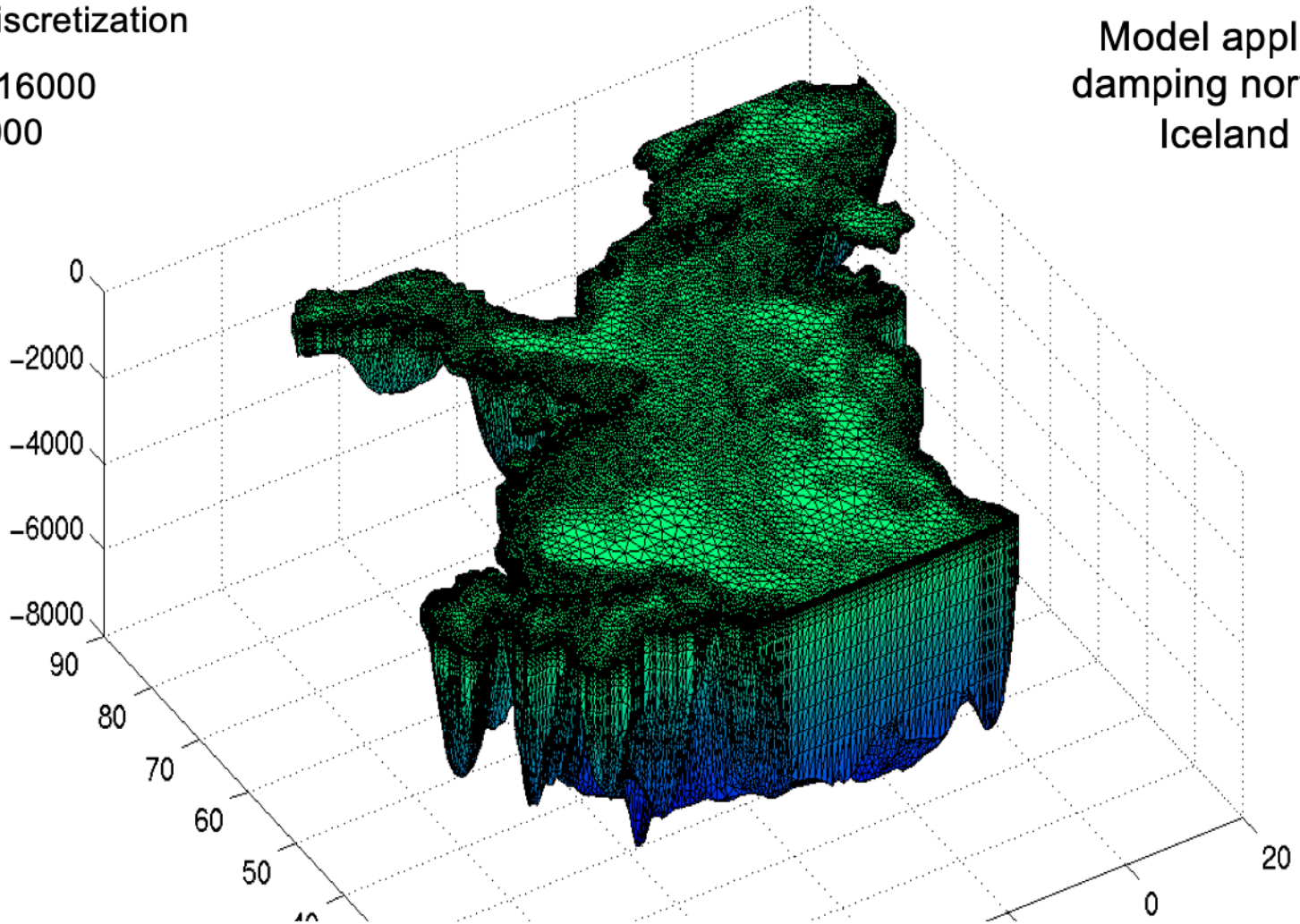
- Possible factors influencing the localization radius
  - ensemble size
  - model dynamics
  - observation correlations

Effect of localization in real model case

FEOM – Coarse mesh for North Atlantic

finite-element discretization

surface nodes: 16000
3D nodes: 220000
z-levels: 23
eddy-permitting



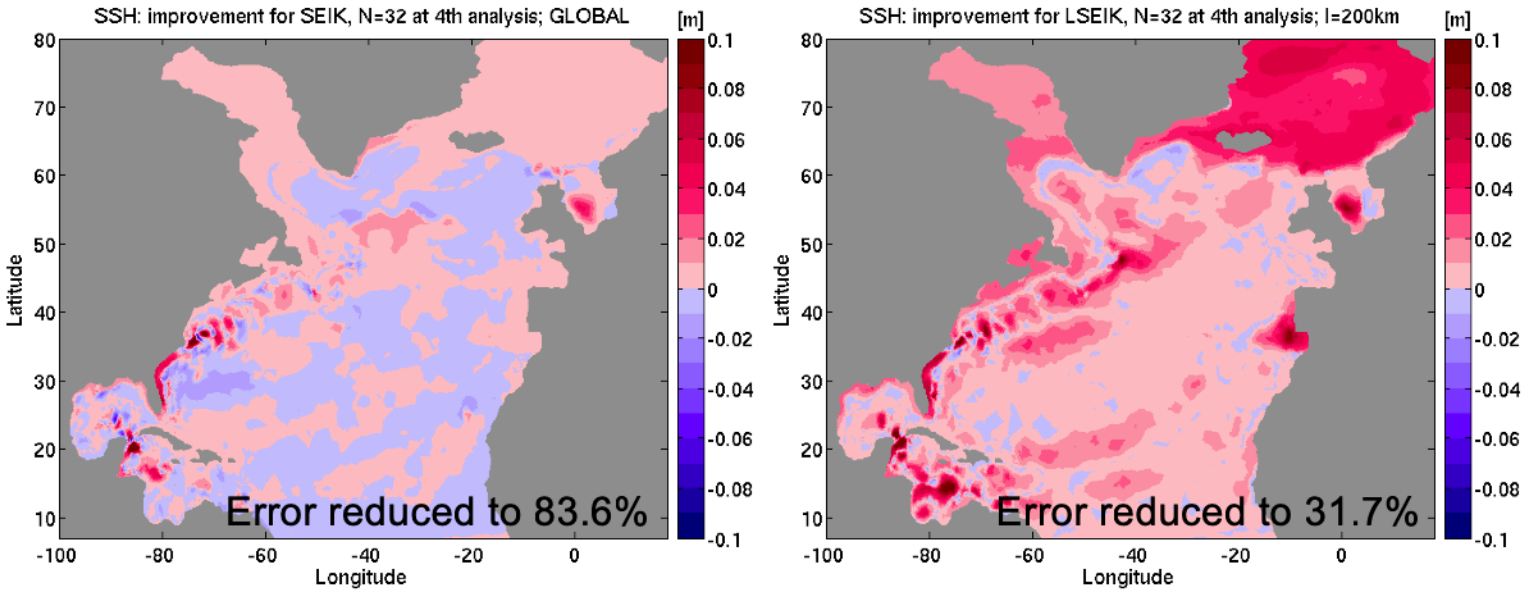
Model applies damping north of Iceland

Configuration of twin experiments

- Generate true state trajectory for 12/1992 - 3/1993
- Assimilate synthetic observations of sea surface height (generated by adding uncorrelated Gaussian noise with std. deviation 5cm to true state)
- Initial Covariance matrix estimated from variability of 9-year model trajectory (1991-1999) initialized from climatology
- Initial state estimate from perpetual 1990 model spin-up
- Monthly analysis updates (at initial time and after each month of model integration)
- No model error; forgetting factor 0.8 for both filters

Nerger, L. et al., J. Mar. Syst. 65 (2007) 288-298

Global vs. local SEIK, N=32 (March 1993)



- *Improvement* is error reduction by assimilation
- Localization extents improvements into regions not improved by global SEIK
- Regions with error increase diminished for local SEIK
- Underestimation of errors reduced by localization

LSEIK: True and estimated errors - third forecast

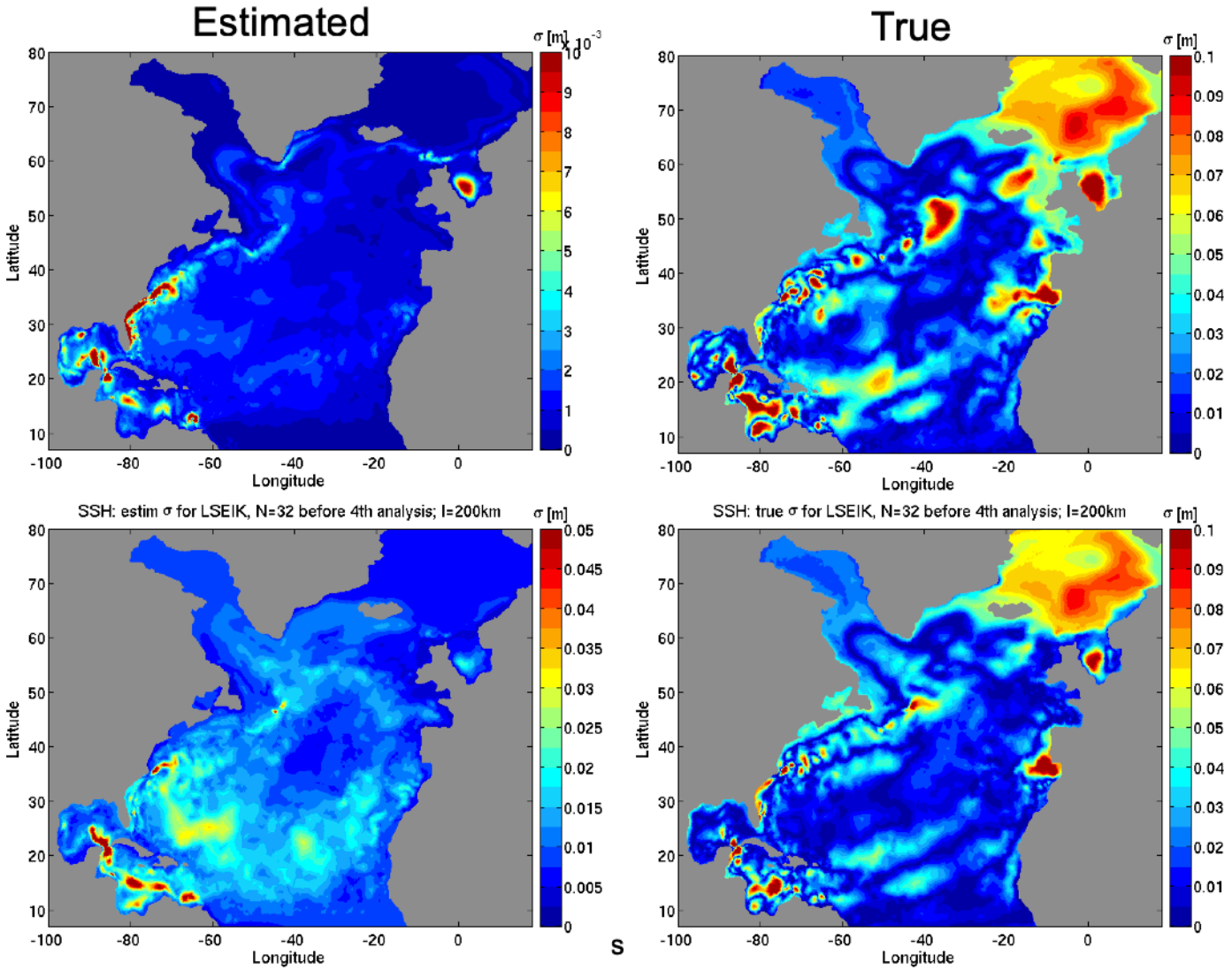
SEIK

Strongly underestimated errors

LSEIK

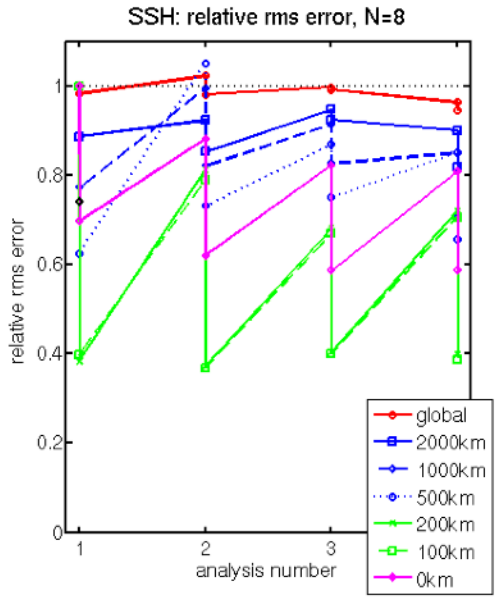
r_l=200km

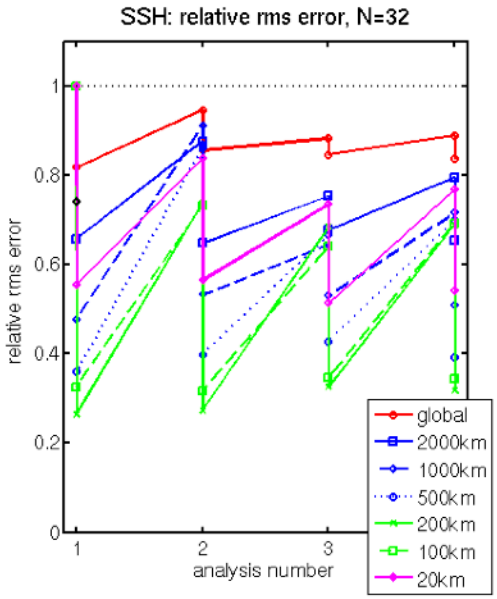
weaker underestimation but no good representation



Relative RMS errors (error reduction) for SSH

N=8





N=32

- global filter: significant improvement for larger ensemble
- global filter with N=100: relative rms error 0.74
- localization strongly improves estimate
  - larger error-reduction at each analysis update
  - but: stronger error increase during forecast
- very small radius results in over-fitting to noise

  ➜ Optimal radius > 0 km

Large-scale Ensemble Data Assimilation

Now we have all we need for large-scale data assimilation

In our applications we use

- ESTKF with
  - Localization
  - Inflation (forgetting factor)
  - Parallelization (another topic…)

all coded in our data assimilation framework PDAF

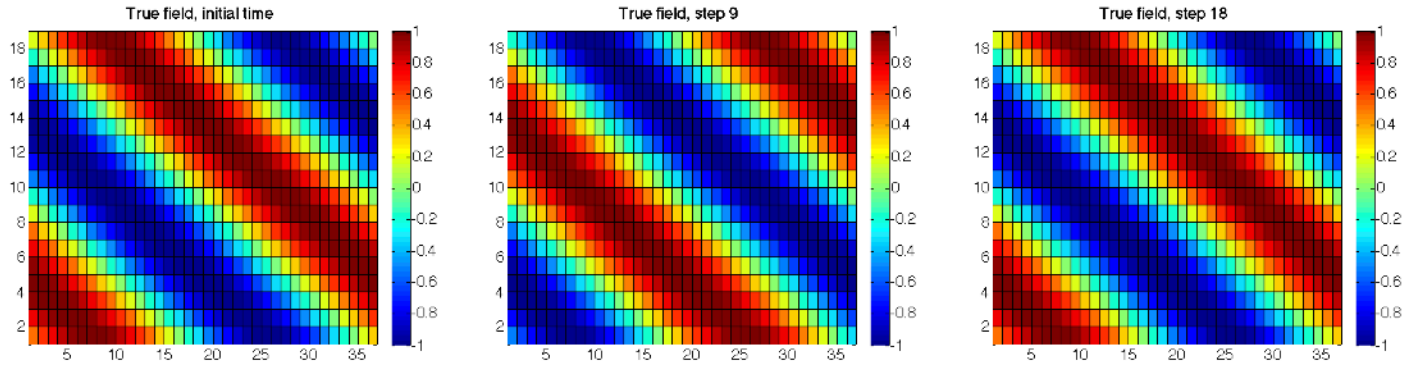
Hands-On Tutorial 3

Ensemble Kalman Filtering with pyPDAF

Hands-On Tutorial 3: Ensemble KF with 2-dimensional test case

Test case:

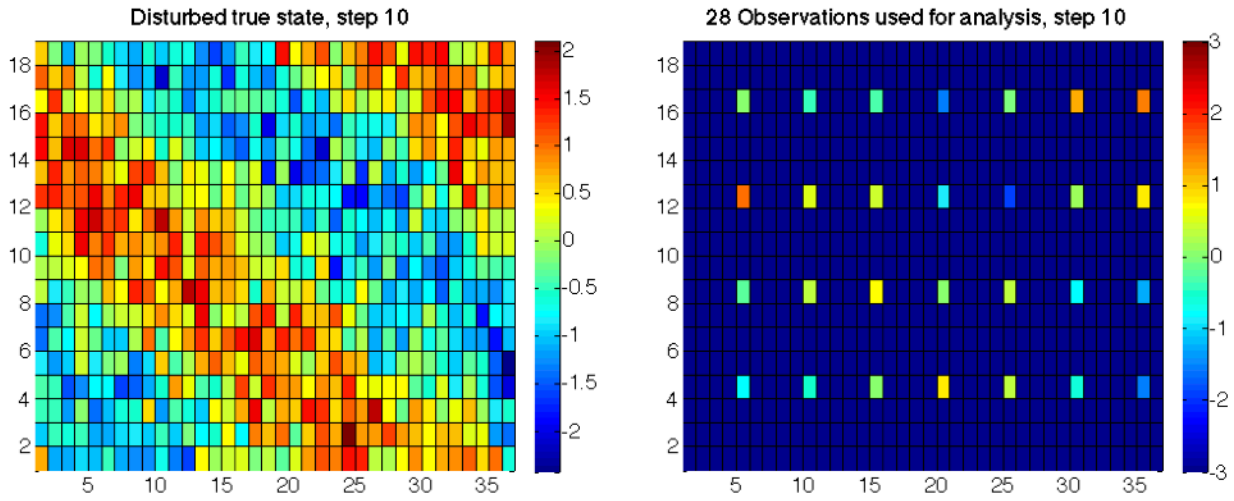
- 2-dimensional rectangular domain
- 36 x 18 grid points (longitude x latitude)
- True state: sine wave in diagonal direction
  (periodic for consistent time stepping)
- Simple time stepping:
  Shift field in vertical direction one grid point per time step
- Output to text files (18 rows) – `true_step*.txt`



Hands-On 3: Ensemble KF with 2-dimensional application

Observations

- Add random error to true state (standard deviation 0.5)
- Select a set of observations at 28 grid points
- File storage (in `inputs_online/`):
  text file, full 2D field, -999 marks 'no data' – `obs_step*.txt`
  one file for each time step



Part 2: Ensemble KF with 2-dimensional application

Using pyPDAF

- pyPDAF is a Python interface to PDAF
  - Model and case-specific parts are coded in Python
  - Actual data assimilation performed in PDAF (using the Fortran code)

Install pyPDAF

- Install pyPDAF from https://github.com/yumengch/pyPDAF
  - following the instructions in README.md

First steps with pyPDAF:

- Download the code from https://github.com/larsnerger/pyPDAF
- The tutorial code is in tutorial/online2D
- The example has to be executed in tutorial/

pyPDAF: General run options

- cd into `tutorial/` and run the program with

  `mpiexec -n X python -u tutorial2D/main.py`

- `X`: ensemble size (between 2 and 9) –has to be set consistently in main.py
- All other `OPTIONS` are specified in the Python code
- Relevant `OPTIONS` for the assignment are in `main.py`:

`dim_ens=4`	ensemble size (between 2 and 9)
`filtertype=4`	select filter algorithm, e.g. 4: global filter ETKF 5: local filter LETKF
`forget=1.0`	select inflation (0 < forget <=1)
`enstype='A'`	select ensemble (A, B, C, D, E)

Note: depending on the computer used you might need to specify `--oversubscribe` with the mpiexec command

Options to run with local filter

- In `tutorial/` run the program with

  `mpiexec -n X python -u tutorial2D/main.py`

- Relevant `OPTIONS` related to localization are

  `Filtertype=5` select LETKF

  `cradius=5.0` set localization cut-off radius
  0.0 by default, any positive value possible

  `locweight=2` set weight function for localization, choices:
  0: constant weight of 1
  1: exponential decrease
  2: decrease with 5^{th} order polynomial (Gaussian-like)

pyPDAF: Plotting files with Python

- Python plot scripts are in `plotting/`
- Plot result of forward model with Python:

```
python plot_file.py true_step18.txt
```

Plot scripts:

- `plot_file.py` – plot the field from a single output file of pyPDAF

```
python plot_file.py FILENAME
```

- `plot_file_input.py` – plot the field from a single file in inputs_online/

```
python plot_file_input.py FILENAME
```

- `plot_diff.py` – plot the difference of two fields (e.g. analysis and truth)

```
python plot_diff.py FILE1 FILE2
```

(the first file has to be a pyPDAF output file and the second from inputs_online/)

- `plot_rmse.py` – plot the rms error of the analysis over time (execute in main directory)

```
python plotting/plot_rmse.py EXPERIMENT
```

Note: pyPDAF outputs with the delimiter=';' while the files in inputs_online/ use blank spaces

Part 1: Assimilation runs with global filter

Here use the code variant in `tutorial2D/` which stores outputs in sub-directories like `out_N4_f1.0/` for `dim_ens=4` and `forget=1.0`

1.1 Run with the ensemble sizes 4 and 9.
Plot the analysis fields and difference of the field to the truth at times (2, 10, 18).

a) How does the difference to the true state change over time?

b) How does the RMSE change over time?

1.2 Vary the forgetting factor for dim_ens=4 in the range 0.1 to 1.0 in steps of 0.1.
Plot the RMSe as a function of the forgetting factor. (You can use the script `plotting/plot_heatmap_forget.py` to be run in main directory)

a) How does the analysis field and its RMSE change at time steps 2 and 18 in dependence on the forgetting factor?

b) How can the dependence on 'forget' be explained?
(Default is forget=1.0; See information on options in pyPDAF on later slides)

Note: The python script plot_diff.py computes also the root mean-square error (RMSE).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\text{field1}(i) - \text{field2}(i))^2}$$

Part 2: Assimilation runs with alternative ensemble

In this task run experiments with the second ensemble type:

- Set `enstype = 'E'`

2.1 Repeat task 1.1

What results do you obtain (fields, difference to truth, RMSE)? How do they compare to the result of task 1.1?

2.2 Repeat task 1.2

What results do you obtain (fields, difference to truth, RMSE)? How do they compare to the result of tass 1.2?

2.3 Plot the ensemble members of the original ensemble and the alternative ensemble (in tutorial/inputs_online/)

How can the results found in 2.1 and 2.2 be explained based on the shape of the ensemble members?

Part 3: Assimilation experiments with localized filter

Here use the code variant in `tutorial2D/` which stores outputs in sub-directories like `out_ensE_N4_lw0_r0.0/` for `enstype='E', locweight=0, cradius=0.0`

- Set `enstype='E'`
- Use the localized filter LETKF with `filtertype=5, forget=1.0, dim_ens=4`
- Run with the additional options

  `cradius` (use {0.0, 1.0, 5.0, 10.0, 15,.0 20.0, 25.0, 30.0, 35.0, 40.0})

  `sradius = cradius`

  `locweight` (use 0 or 2)

a) How does the analysis field at times 2 and 18 change when varying `cradius` for `locweight=0`?

b) How does the analysis field at times 2 and 18 change when varying `cradius` for `locweight=2`? How to the results compare with those from task a?

c) For both choices of `locweight` find a choice of `cradius` for which the RMSE is minimized at times 2 and 18? Why does this choice lead to the best result?

Note: The script `plotting/plot_heatmap_local.py` allows to get an easy overview)

Part 4: Assimilation runs with original ensemble

In this task run experiments with the original ensemble type
which gave the best results for the global filter:

- Set `enstype='A'`

a) For both choices of `locweight` find a choice of `cradius`
for which the RMSE is minimized. How far does it differ from that in 1.1c?
Why does this choice might lead to be best result?