NMEFC, Beijing, China, November 9, 2017

High-Dimensional Nonlinear Data Assimilation with the Nonlinear Ensemble Transform Filter (NETF) and its Smoother Extension

Lars Nerger, Paul Kirchgessner,

Alfred Wegener Institute, Bremerhaven, Germany

Julian Tödter, Bodo Ahrens

University of Frankfurt, Frankfurt, Germany

SANGOMA

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

Overview

- Study new Nonlinear Ensemble Transform Filter – NETF (Tödter & Ahrens, MWR, 2015)
- Extend NETF for smoothing
- Test filter and smoother in realistic high-dimensional idealized ocean data assimilation experiments

Kalman and Nonlinear Filters

Ensemble filters – ensemble Kalman filters & NETF

- represent state and its error by ensemble $\mathbf{X}$ of N states
- Forecast:
  - Integrate ensemble with numerical model
- Analysis:
  - update ensemble mean $\overline{\mathbf{x}}^a = \overline{\mathbf{x}}^f + \mathbf{X}'^f \tilde{\mathbf{w}}$
  - update ensemble perturbations $\mathbf{X}'^a = \mathbf{X}'^f \mathbf{W}$

  (both can be combined in a single step)
- Ensemble Kalman filters & NETF: Different definitions of
  - weight vector $\tilde{\mathbf{w}}$
  - Transform matrix $\mathbf{W}$

The Ensemble Kalman Filter (EnKF, Evensen 94)

Ensemble $\{\mathbf{x}_0^{a(l)}, l = 1, \ldots, N\}$

Ensemble covariance matrix

$$\mathbf{P}_k^f := \frac{1}{N-1} \sum_{l=1}^{N} \left(\mathbf{x}_k^{f(l)} - \overline{\mathbf{x}_k^f} \right) \left(\mathbf{x}_k^{f(l)} - \overline{\mathbf{x}_k^f} \right)^T$$

Ensemble mean (state estimate)

$$\mathbf{x}_k^a := \frac{1}{N} \sum_{l=1}^{N} \mathbf{x}_k^{a(l)}$$

Analysis step:

Update each ensemble member

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

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

Kalman filter

Expensive to compute

moother

Efficient use of ensembles

Kalman gain

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

Alternative form (Sherman-Morrison-Woodbury matrix identity)

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

Looks worse: $n \times n$ matrices need inversion

However: with ensemble $\mathbf{P}_k^f = (N-1)^{-1} \mathbf{X}' \mathbf{X}'^T$

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

Inversion of $N \times N$ matrix

(Ensemble perturbation matrix $\mathbf{X}' = \mathbf{X} - \bar{\mathbf{X}}$)

moother

ETKF (Bishop et al., 2001)

- Ensemble Transform Kalman filter:
  - Transform matrix

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

  - Mean update weight vector

$$\tilde{\mathbf{w}} = \mathbf{A}(\mathbf{H}\mathbf{X}'^{f})^{T}\mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{H}\overline{\mathbf{x}^{f}}\right)$$

    (depends on $\mathbf{R}$ and $\mathbf{y}$)

  - Transformation of ensemble perturbations

$$\mathbf{W} = \sqrt{(N-1)}\mathbf{A}^{-1/2}\mathbf{\Lambda}$$

    (depends only on $\mathbf{R}$, not $\mathbf{y}$)

Particle filters – fully nonlinear ensemble filters

- Avoid changing ensemble members ('particles')
- Instead: give particles a weight at change it at the analysis step
  - Initial weight: 1/N for all particles
- Weights are given by statistical likelihood of an observation
- Example: With Gaussian observation errors (for each particle *i*):

$$\tilde{w}^i \sim \exp\left(-0.5(\mathbf{y}-\mathbf{H}\mathbf{x}_i^f)^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{H}\mathbf{x}_i^f)\right)$$

- Ensemble mean state computed with weights

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

- This update does not assume any distribution of the state errors (and is not limited to Gaussian distributations)

Nonlinear ensemble transform filter - NETF

- Ensemble Kalman:
  - Transformation according to KF equations
- NETF (Tödter & Ahrens, MWR, 2015)
  - Mean update from Particle Filter weights: for all particles *i*

$$\tilde{w}^i \sim \exp\left(-0.5(\mathbf{y}-\mathbf{H}\mathbf{x}_i^f)^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{H}\mathbf{x}_i^f)\right)$$

  - Ensemble update
    - Transform ensemble to fulfill analysis covariance (like KF, but not assuming Gaussianity)
    - Derivation gives

$$\mathbf{W} = \sqrt{N}\left[\mathrm{diag}(\tilde{\mathbf{w}}) - \tilde{\mathbf{w}}\tilde{\mathbf{w}}^T\right]^{1/2}\Lambda$$

($\mathbf{\Lambda}$: mean-preserving random matrix; useful for stability)

(Almost same formulation: Xiong et al., Tellus, 2006)

Derivation of NETF

- Mean state update

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

- Analysis covariance matrix

$$\mathbf{P}^a = \sum_{i=1,N} \tilde{w}_i \, (\mathbf{x}_i^f - \overline{\mathbf{x}}^a)(\mathbf{x}_i^f - \overline{\mathbf{x}}^a)^T$$

$$\mathbf{P}^a = \frac{1}{N} \mathbf{X}^f \mathbf{W}^2 (\mathbf{X}^f)^T$$

with

$$\mathbf{W} = \sqrt{N} \left[\mathrm{diag}(\mathbf{w}) - \tilde{\mathbf{w}} \tilde{\mathbf{w}}^T \right]^{1/2} \mathbf{\Lambda}$$

Difference of ETKF and NETF

- ETKF parameterizes ensemble distribution by a Gaussian distribution
- NETF uses particle filter weights to ensure correct update of ensemble mean and covariance

- Filter update:
  - in ETKF is linear in observations

$$\tilde{\mathbf{w}} = \mathbf{A}(\mathbf{H}\mathbf{X}'^{f})^{T}\mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{H}\overline{\mathbf{x}^{f}}\right)$$

  - in NETF is nonlinear in observations

$$\tilde{w}^{i} \sim \exp\left(-0.5(\mathbf{y} - \mathbf{H}\mathbf{x}_{i}^{f})^{T}\mathbf{R}^{-1}(\mathbf{y} - \mathbf{H}\mathbf{x}_{i}^{f})\right)$$

Ensemble Smoothers – ETKS & NETS

- Smoother: Update past ensemble with future observations
- Rewrite ensemble update as
  - Filter:

$$\mathbf{X}^a_{k|k} = \mathbf{X}^f_{k|k-1}\hat{\mathbf{W}}_k$$

analysis time

Observations used up to time

- Smoother at time $i < k$

$$\mathbf{X}^a_{i|k} = \mathbf{X}^f_{i|k-1}\hat{\mathbf{W}}_k$$

➢ works likewise for ETKS and NETS

➢ also possible for localized filters

See, e.g., Nerger, Schulte & Bunse-Gerstner, QJRMS 140 (2014) 2249–2259

Experiments with small Lorenz-96 model

Configuration of Lorenz-96 model experiments

Lorenz-96:

- 1-dimensional period wave
- Chaotic dynamics

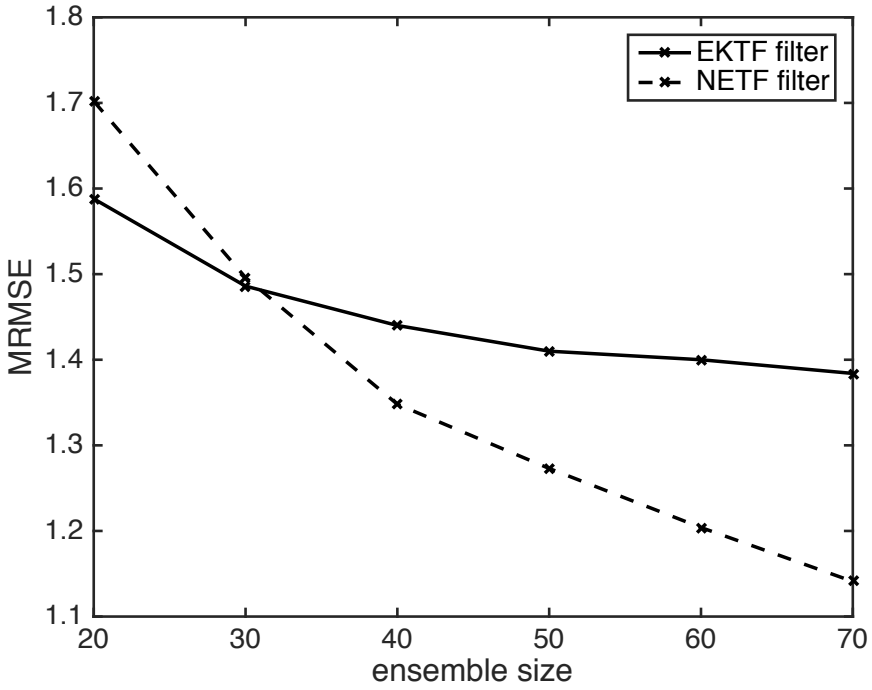
Configuration for assimilation experiments

- State dimension: 80
- Observed: 40 grid points
- Time steps between analysis steps: 8
- Double-exponential observation errors
  (for even stronger nonlinearity)
- Experiment length: 5000 time steps
- Observation error standard deviation: 1

  ➜ this is a difficult case for the assimilation

Performance of NETF – Lorenz-96

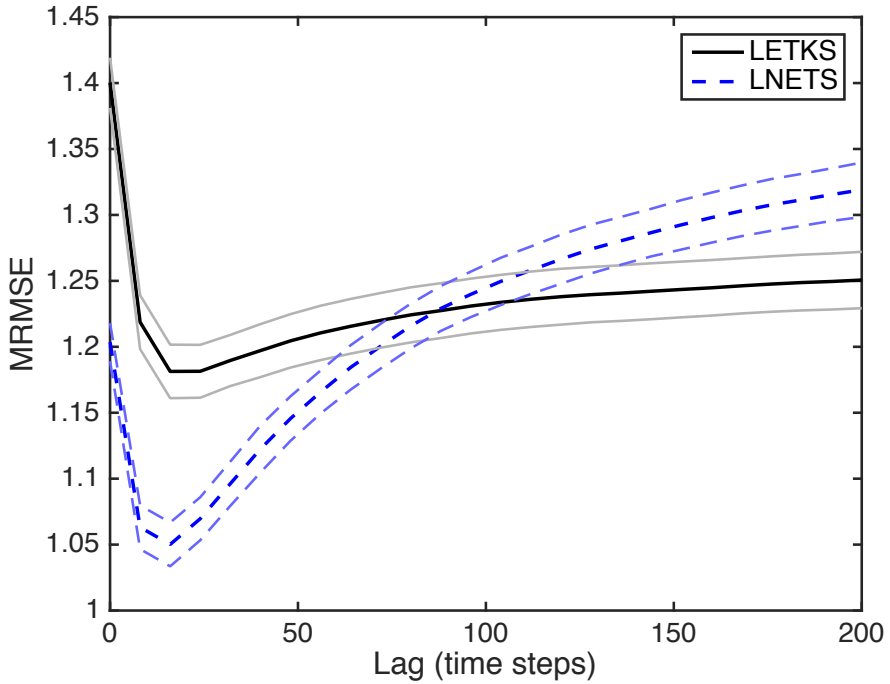
- Performance for small model (Lorenz-96)



- NETF beats ETKF for ensemble size larger 30

Appliction of smoother

- Time period over which smoothing is performed: smoother lag



Typical behavior with nonlinear models

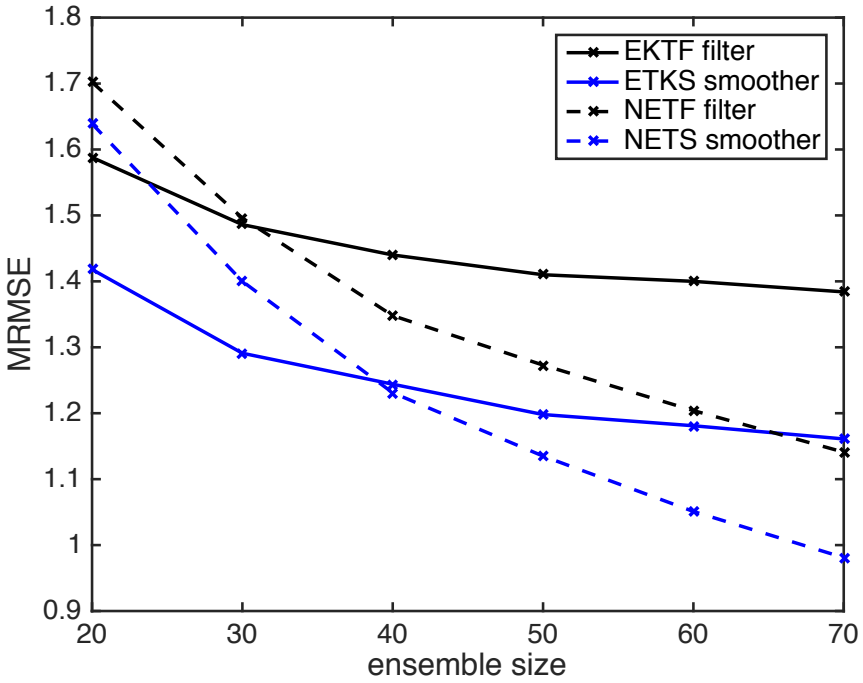
- Fast reduction of error short lag
- Error increase for large lag (caused by nonlinarity)

  ➜ There is an optimal lag with minimum error

L. Nerger, S. Schulte, A. Bunse-Gerstner (2014) QJR. Meteorol. Soc. 140: 2249

Performance of NETF – Lorenz-96

- Performance for small model (Lorenz-96)



- Blue: Smoother
- NETS beats ETKS for ensemble size 40 and larger
  - Smoother slightly stronger for ETKS
  - NETS better than ETKF filter for N=70

Kirchgessner, Toedter, Ahrens, Nerger. (2017) Tellus A 69:1, 1327766

NETF/NETS

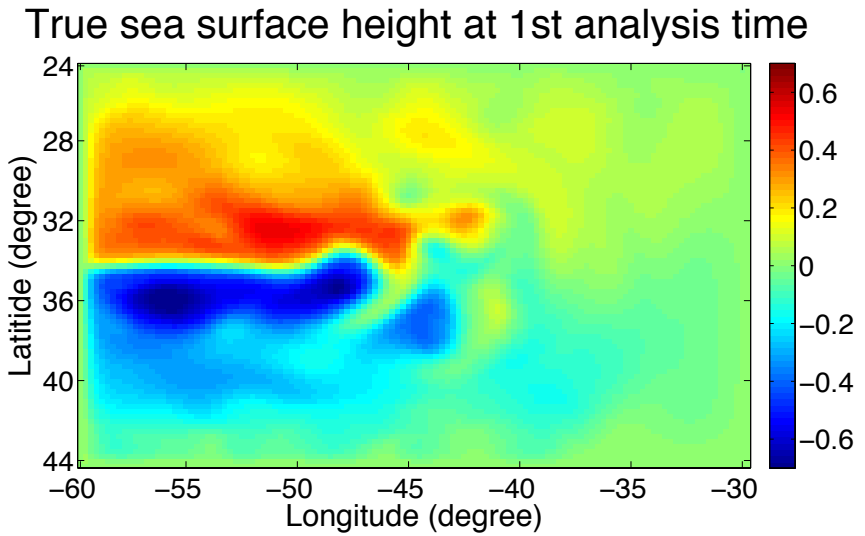
with
high-dimensional ocean model

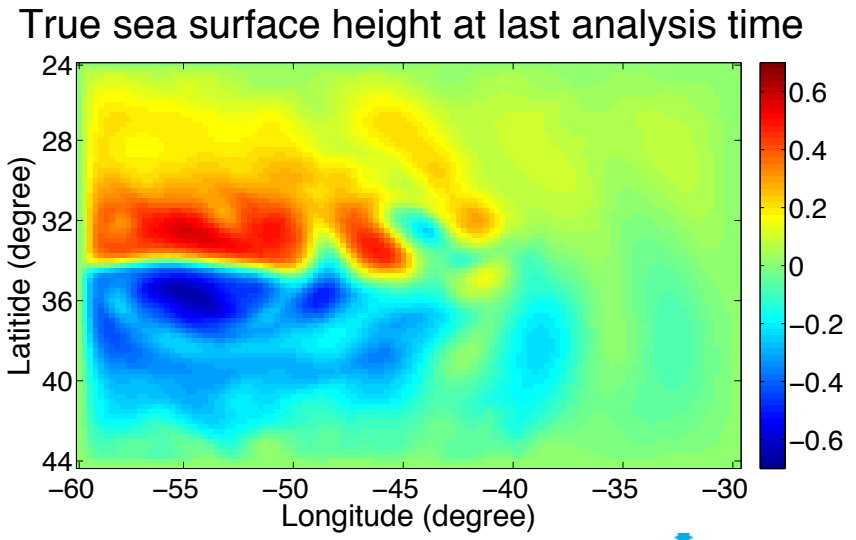
Assimilation into NEMO

European ocean circulation model

Model configuration

- box-configuration “SEABASS”
- ¼° resolution
- 121x81 grid points, 11 layers (state vector ~300,000)
- wind-driven double gyre (a nonlinear jet and eddies)
- medium size SANGOMA benchmark





SANGOMA

www.data-assimilation.net

AWI

PDAF: A tool for data assimilation

PDAF Parallel Data Assimilation Framework

PDAF - Parallel Data Assimilation Framework

- a program library for ensemble data assimilation
- provide support for parallel ensemble forecasts
- provide fully-implemented & parallelized filters and smoothers (EnKF, LETKF, NETF, EWPF … easy to add more)
- easily useable with (probably) any numerical model (applied with NEMO, MITgcm, FESOM, HBM, TerrSysMP, …)
- run from laptops to supercomputers (Fortran, MPI & OpenMP)
- first public release in 2004; continued development
- ~250 registered users; community contributions

Open source:
Code, documentation & tutorials at

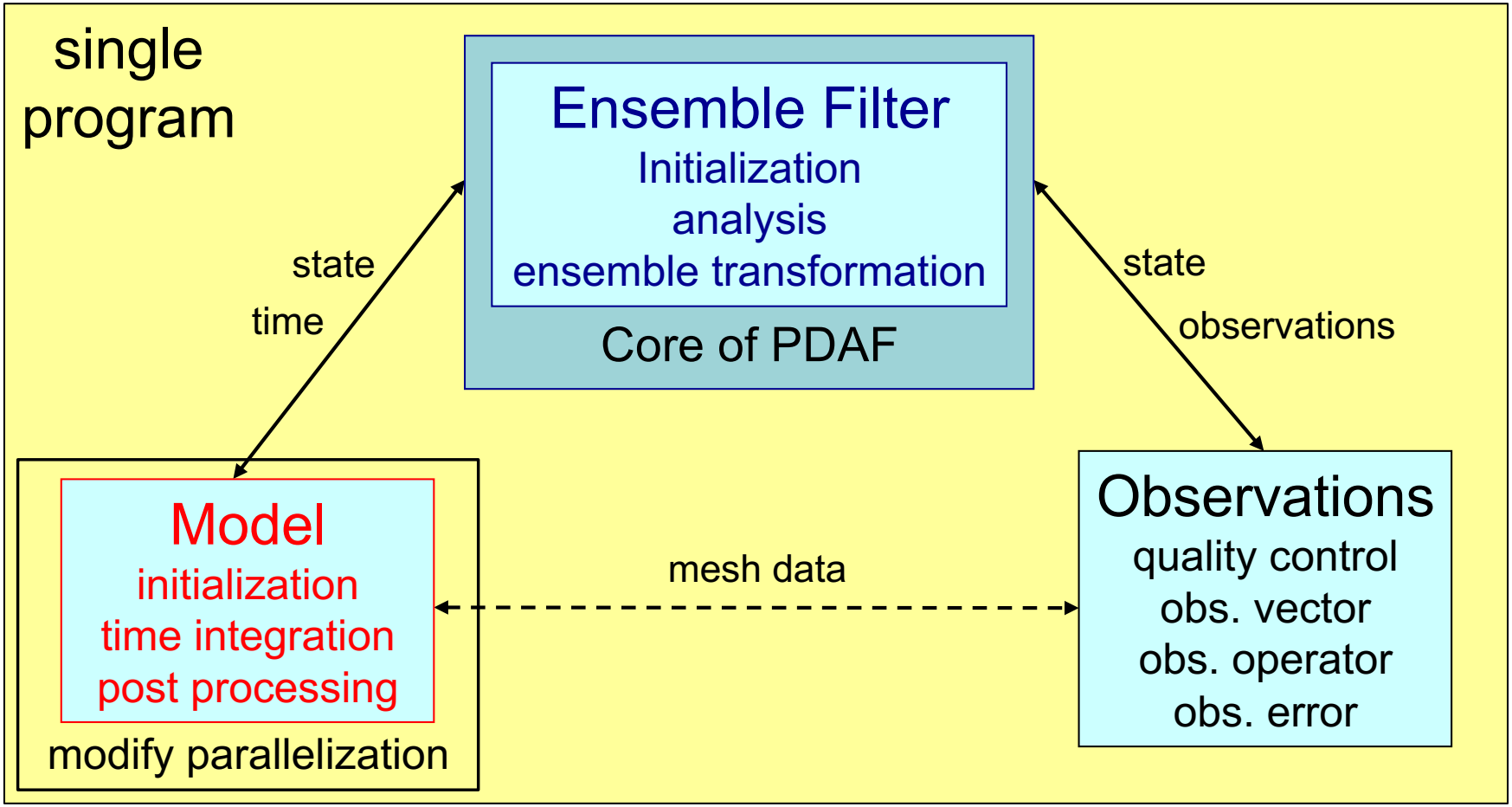
http://pdaf.awi.de

L. Nerger, W. Hiller, Computers & Geosciences 55 (2013) 110-118

AWI

Logical separation of assimilation system

PDAF Parallel Data Assimilation Framework



Explicit interface

Indirect exchange (module/common)

Nerger, L., Hiller, W. Software for Ensemble-based DA Systems – Implementation and Scalability. Computers and Geosciences 55 (2013) 110-118

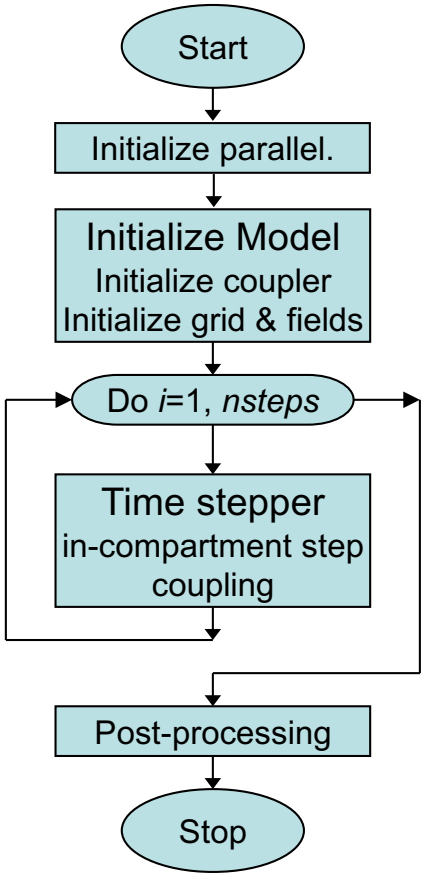
AWI

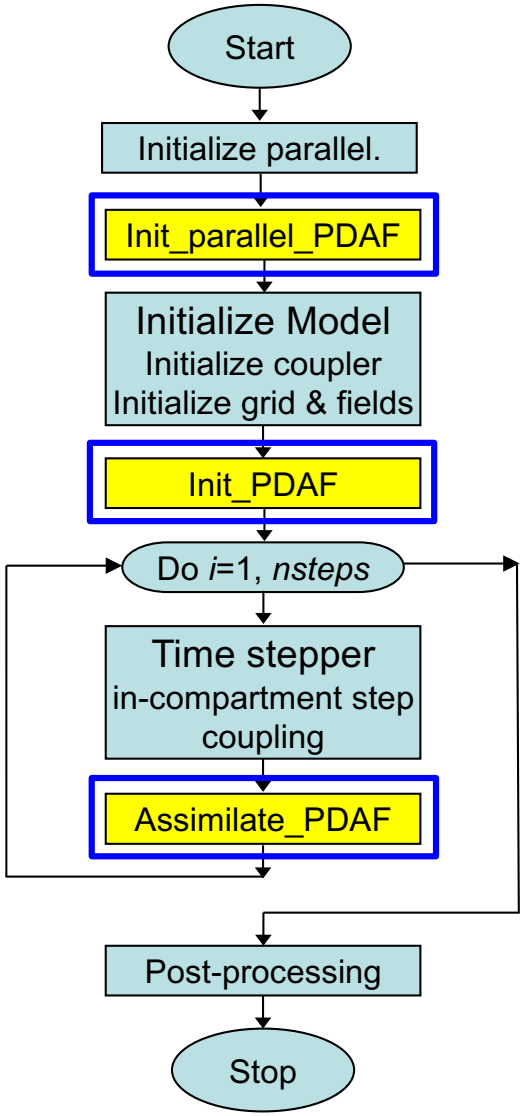
Extending a Model for Data Assimilation

Model

single or multiple executables

coupler might be separate program





Extension for data assimilation

plus:
Possible model-specific adaption:

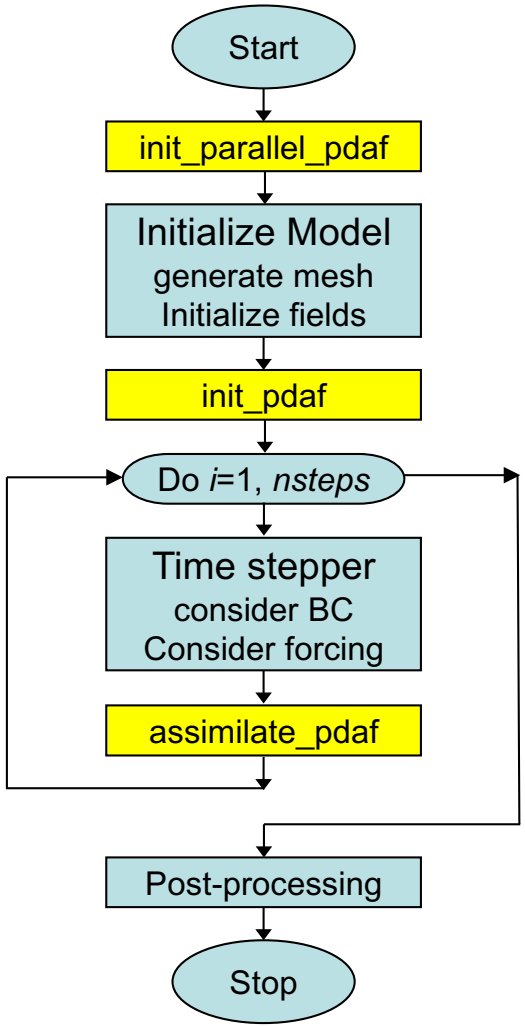
for NEMO: handle leapfrog time stepping

revised parallelization enables ensemble forecast

Features of online-coupled DA program

PDAF Parallel Data Assimilation Framework

- minimal changes to model code when combining model with filter algorithm
- model not required to be a subroutine
- no change to model numerics!
- model-sided control of assimilation program (user-supplied routines in model context)
- observation handling in model-context
- filter method encapsulated in subroutine
- complete parallelism in model, filter, and ensemble integrations



Online coupling: Minimal changes to NEMO

Add to *mynode* (lin_mpp.F90) just before init of myrank

```
#ifdef key_USE_PDAF
   CALL init_parallel_pdaf(0, 1, mpi_comm_opa)
#endif
```

Add to *nemo_init* (nemogcm.F90) at end of routine

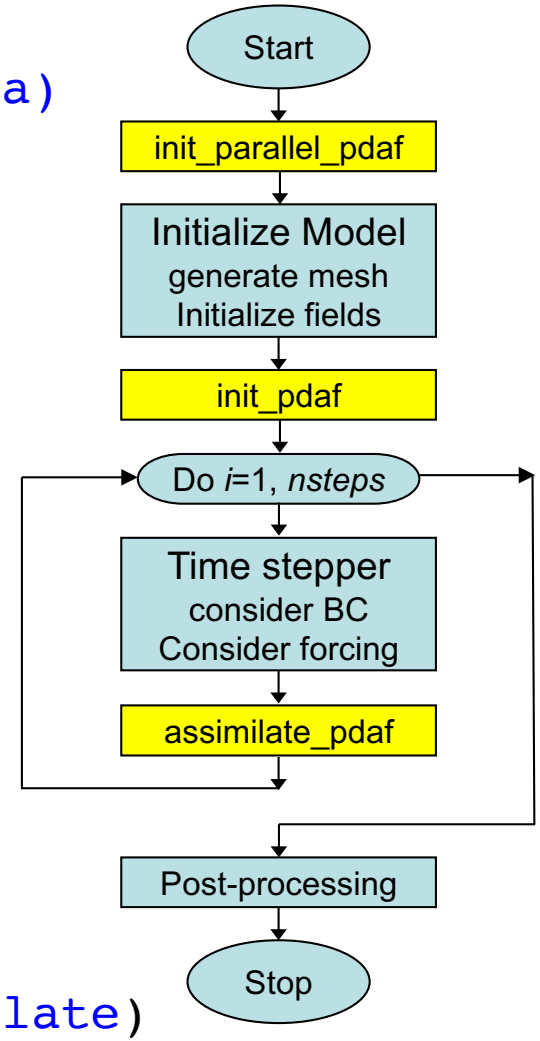
```
#ifdef key_USE_PDAF
   CALL init_pdaf()
#endif
```

Add to *stp* (step.F90) at end of routine

```
#ifdef key_USE_PDAF
   CALL assimilate_pdaf()
#endif
```

Modify *dyn_nxt* (dynnxt.F90)

```
#ifdef key_USE_PDAF
   IF((neuler==0 .AND. kt==nit000).OR.assimilate)
#else
```



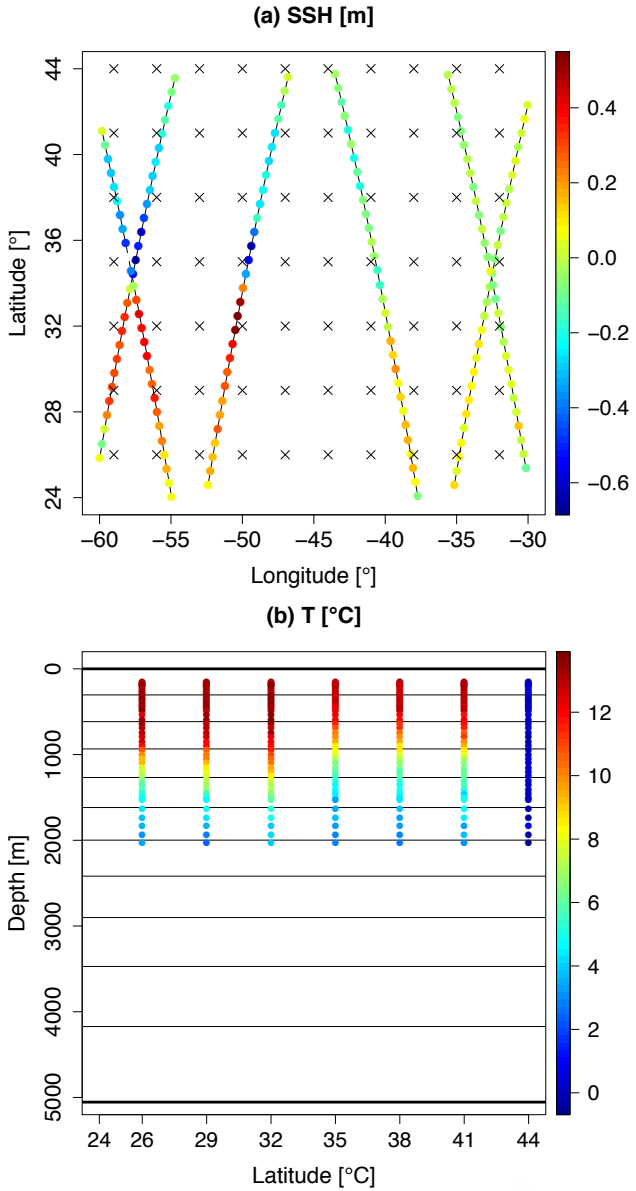
Observations and Assimilation Configuration

Observations

- Simulated satellite sea surface height SSH (Envisat & Jason-1 tracks), 5cm error
- Temperature profiles on 3°x3° grid, surface to 2000m, 0.3°C error

Data Assimilation

- Ensemble size 120
- LETKF, LNETF and smoothers
- Localization: weights on matrix $\mathbf{R}^{-1}$ (Gaspari/Cohn'99 function, 2.5° radius)
- Assimilate each 48h over 360 days

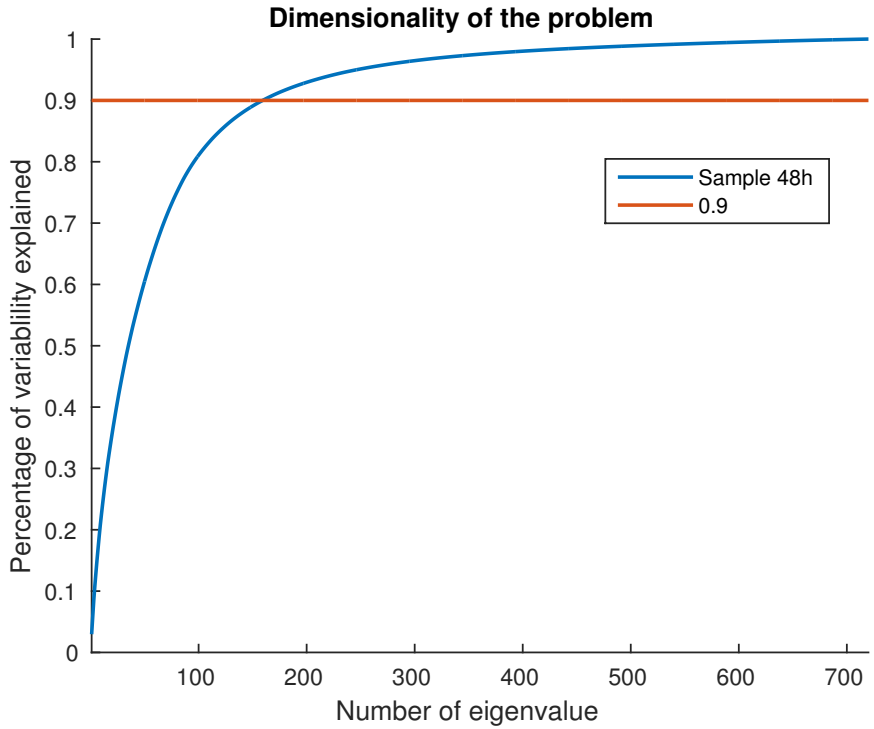


Dimensions of the problem

State vector dimension ~300,000

Dimension of dynamics (error space):

From eigenvalue decompositions (EOFs)

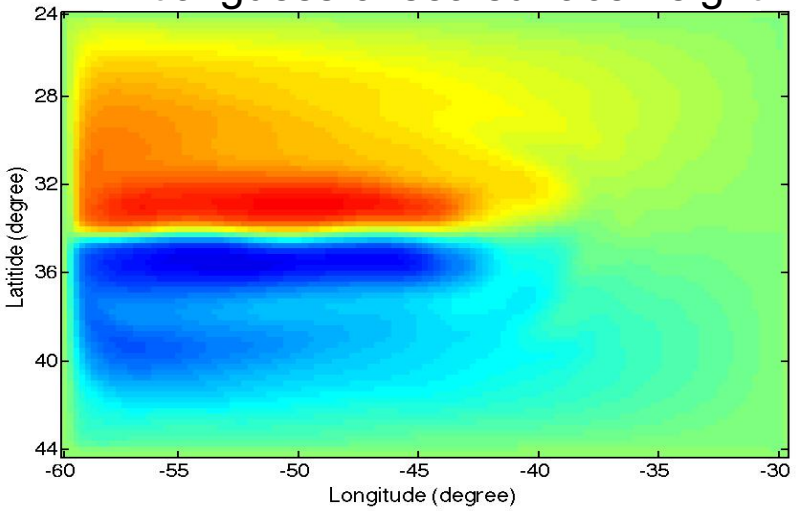


~180 modes for 90% of variability

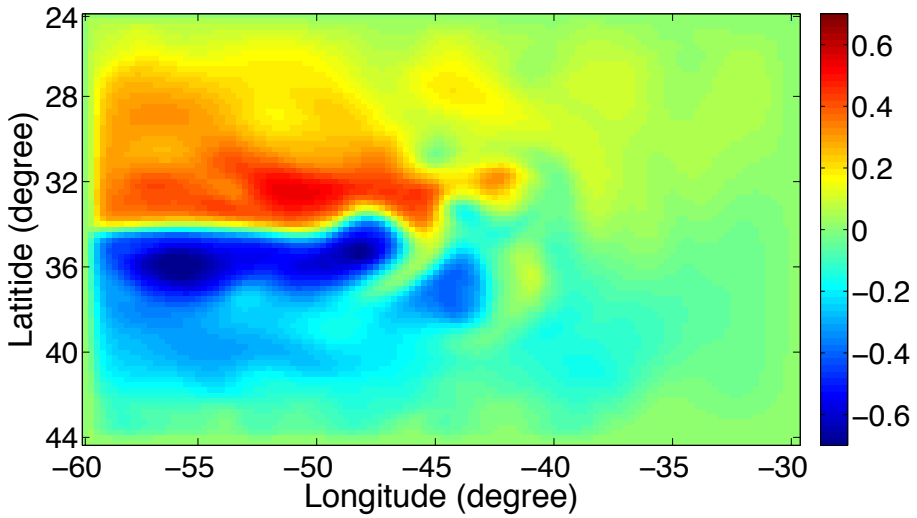
~400 modes for 99.9% of variability

Application of LETKF

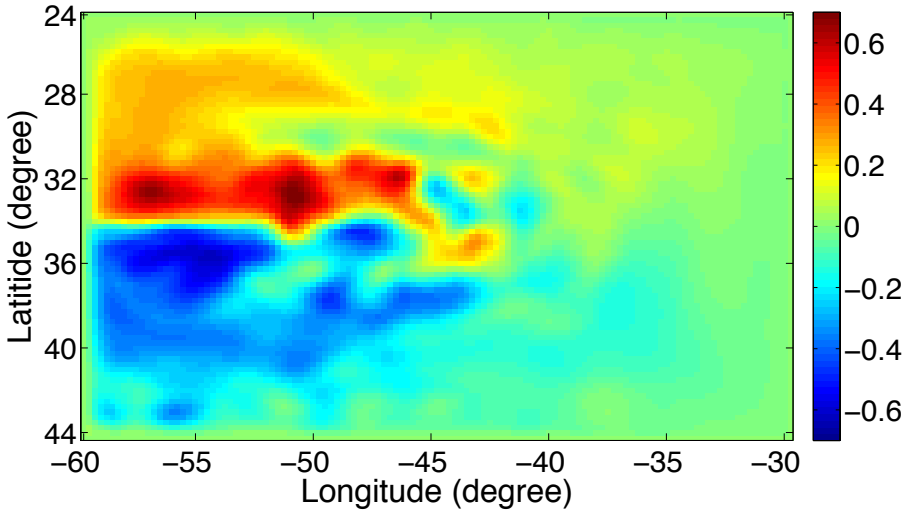
Initial guess of sea surface height



True sea surface height at 1st analysis time



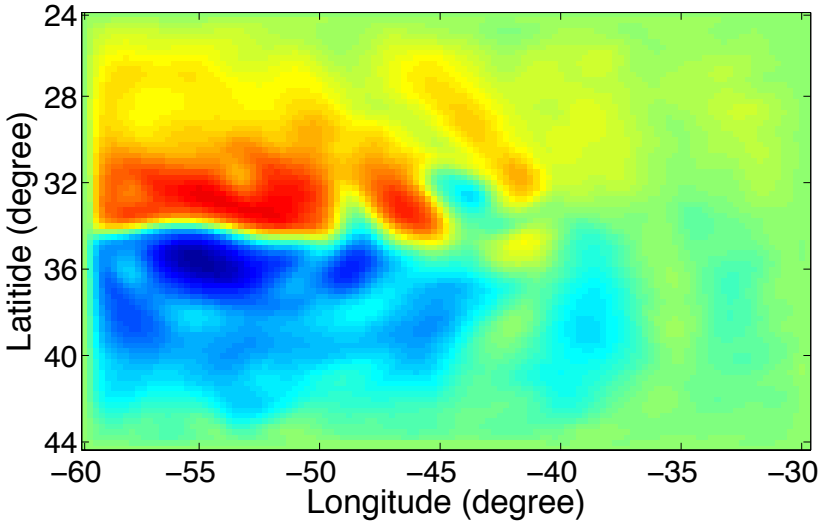
Estimated SSH at 1st analysis time



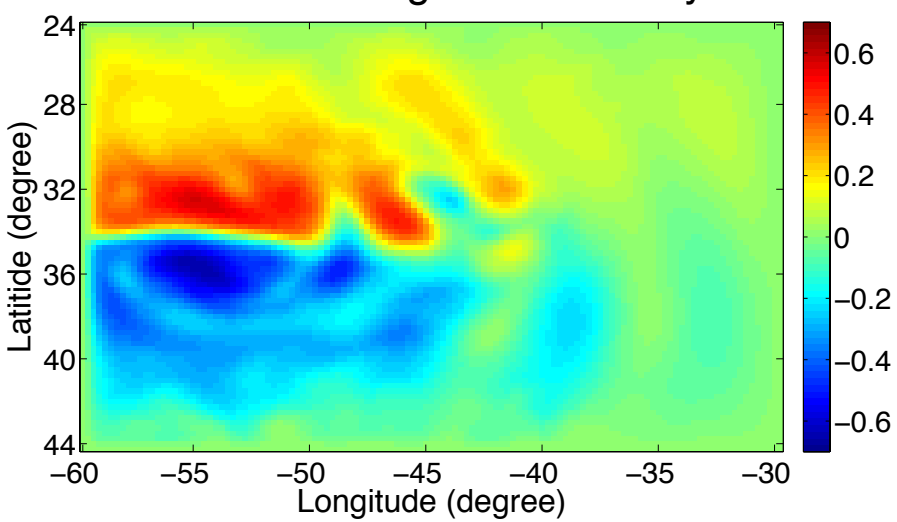
Nonlinear Ensemble Transform Filter & Smoother

Application of LETKF (2)

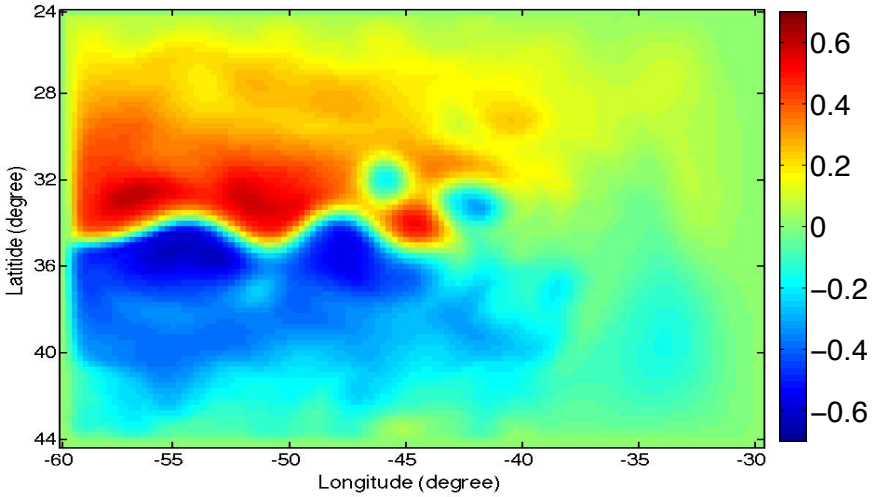
Estimated SSH at last analysis time



True sea surface height at last analysis time



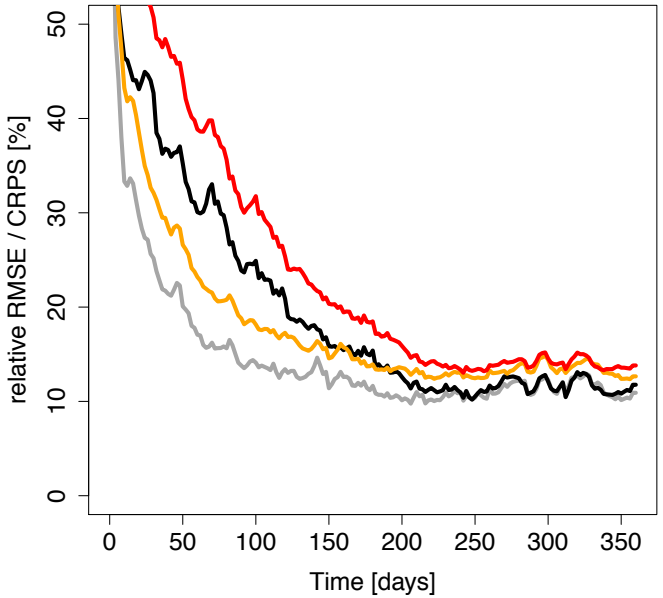
Final SSH without assimilation



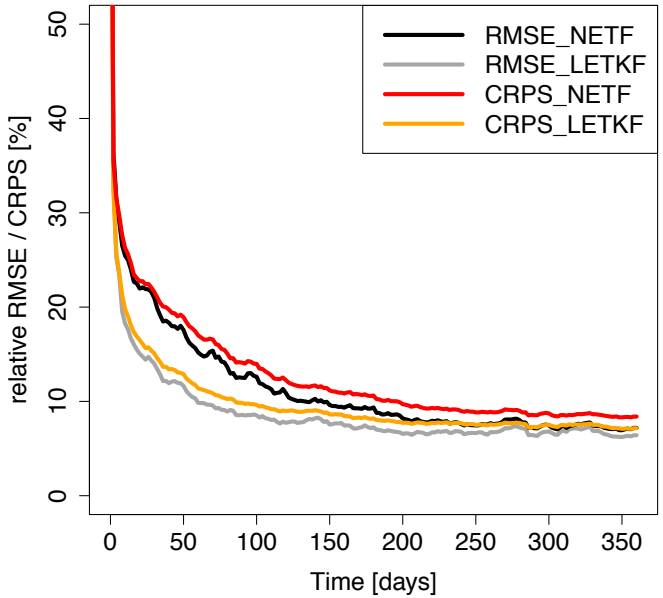
Filter performances in NEMO

- RMS errors reduced to 10% (velocities to 20%) of initial error
- Slower convergence for NETF, but to same error level as LETKF
- CRPS (Continuous Rank Probability Score) shows similar behavior

SSH: Relative error reduction



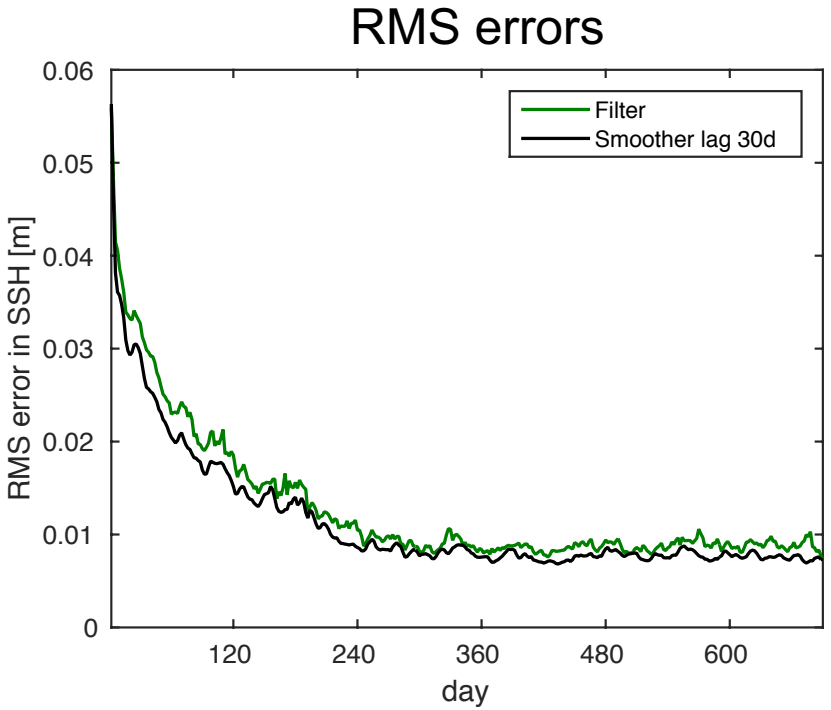
T: Relative error reduction

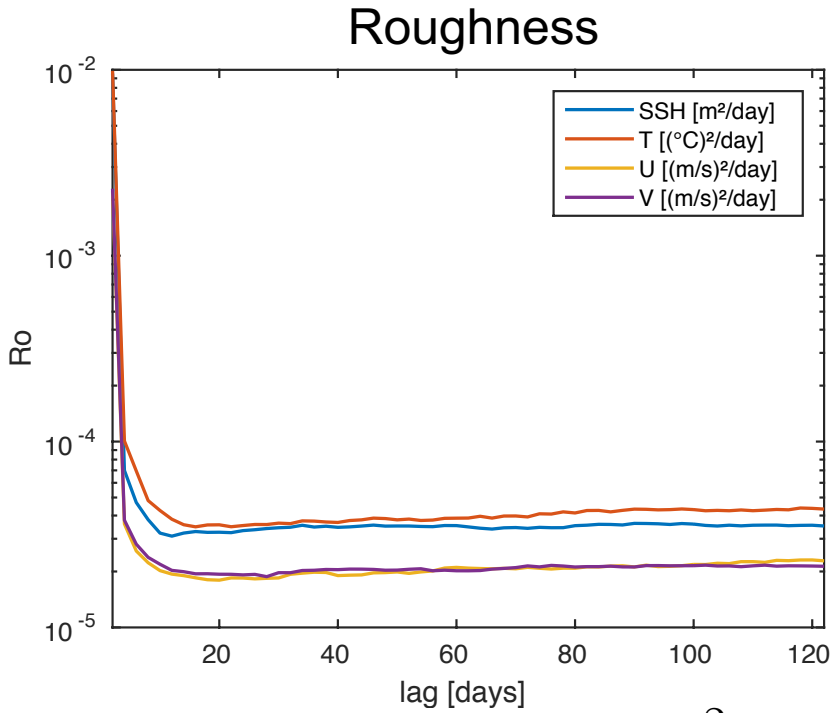


Tödter, Kirchgessner, Nerger & Ahrens, MWR 144 (2016) 409 – 427

Applying the smoother

- Smoother reduces filter errors by ~10%
- Can be useful as smoothing is cheap to compute
- Roughness of error trajectory is strongly reduced (smoothed)





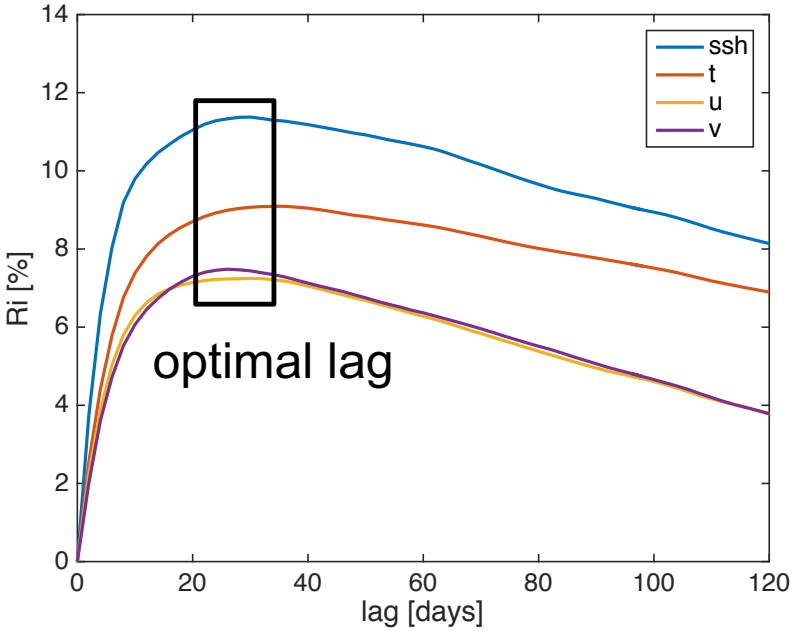
$$Ro = \int \left(\frac{dRMSE}{dt} \right)^2 dt$$

Different smoothing impact

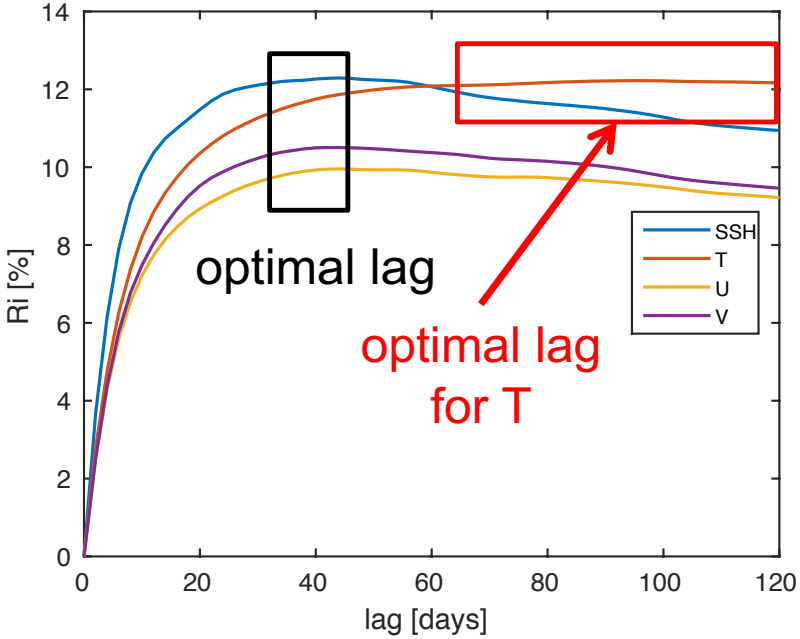
- Consider relative improvement

$$\mathrm{Ri} = 100 \cdot \left(1 - \frac{\mathrm{RMSE}_{smoother}}{\mathrm{RMSE}_{filter}}\right)$$

L-NETF



LETKF

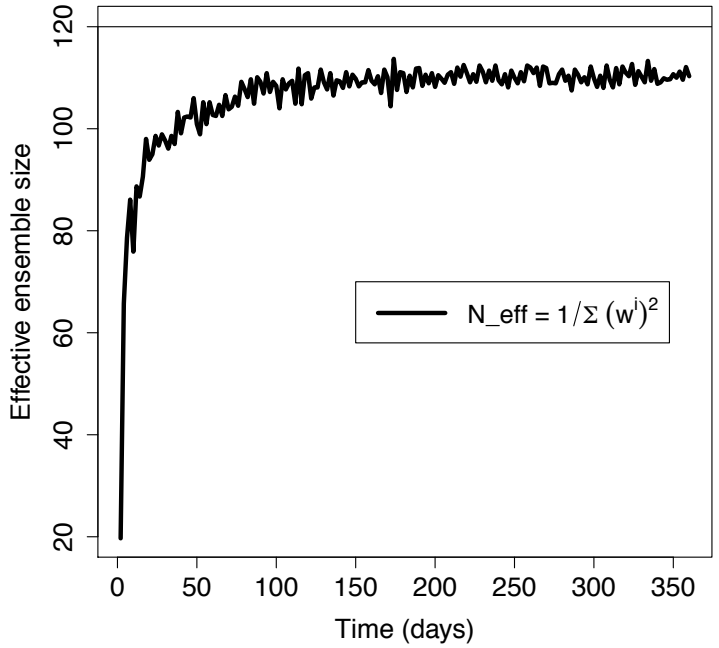


- Similar behavior for ssh (sea surface height)
- Distinct for T
  - ➢ Effect of distinct update schemes (NETF uses observation values for both state and ensemble update)

Ensemble Quality

Effective ensemble size

$$N_{eff} = \frac{1}{\sum_{i=1,N} (w^i)^2}$$

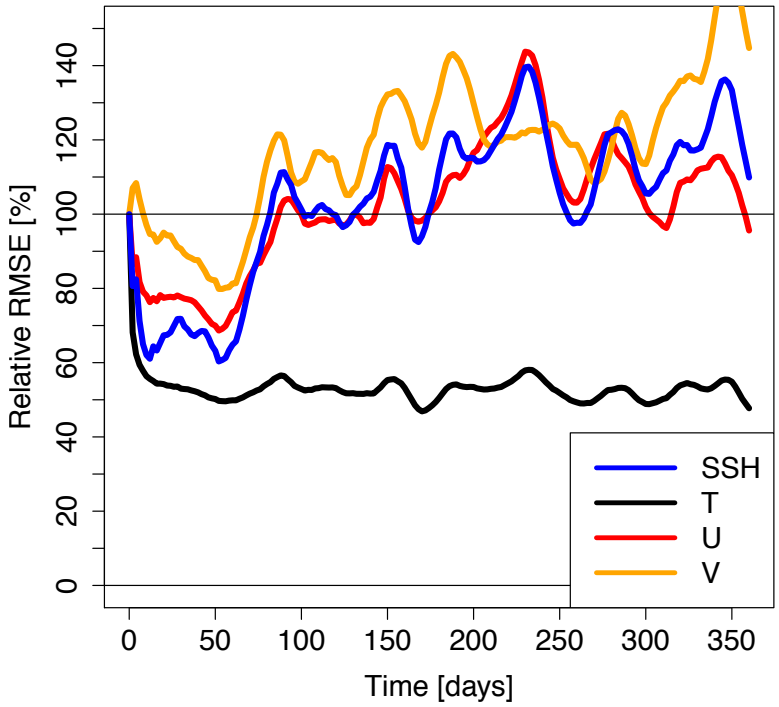


- shows inequality of ensemble weights
- Particle filters need variation in ensemble weights to work
- Extreme cases
  - N_{eff} = 1 (one particle has all weight; degenerate ensemble)
  - N_{eff} = N (all particles has same weight)
- Experiment
  - Strongly variable weights at beginning
  - Lower variability later (but N_{eff} < N)

Tödter, Kirchgessner, Nerger, Ahrens, MWR 144 (2016) 409 – 427

Sensitivity to initial ensemble

- NETF is sensitive to initial ensemble choice
- Using a non-representative ensemble
  (from model variability over 10 years)



Tödter, Kirchgessner, Nerger, Ahrens, MWR 144 (2016) 409 – 427

Summary

- Nonlinear ensemble transform filter/smoother (NETF/S)
  - Update state estimate as particle filter
  - Transform ensemble using covariance matrix
- NEMO ocean test case
  - NETF filtering performance similar to LETKF
  - Slower convergence
  - Sensitive on ensemble size
  - Successful smoothing
    - Dependence on lag distinct for LETKS & NETS (due to different update schemes)

Thank you!