EGU General Assembly 2019

Short Course SC1.1

Data assimilation in the geosciences –

Practical data assimilation with the Parallel Data Assimilation Framework

Lars Nerger[1], Maria Broadbridge[2],
Gernot Geppert[3], Peter Jan van Leeuwen[2,4]

1: Alfred Wegener Institute, Bremerhaven, Germany

2: University of Reading, UK

3: Deutscher Wetterdienst (DWD), Offenbach, Germany

4: Colorado State University, Fort Collins, CO, USA

PDAF Parallel Data Assimilation Framework

The Short Course – Overview

1. Introduction to ensemble data assimilation
2. Implementation concept of PDAF
   (Parallel Data Assimilation Framework)
3. Hands-on Example:
   Build an Assimilation System with PDAF

1

Introduction to

Ensemble Data Assimilation

Overview

- What can we expect to achieve with data assimilation?
- What do we need for data assimilation?
- How does ensemble data assimilation work?
- How can we apply ensemble data assimilation?

Please note:
We omit equations of assimilation methods because you can apply PDAF without knowing them

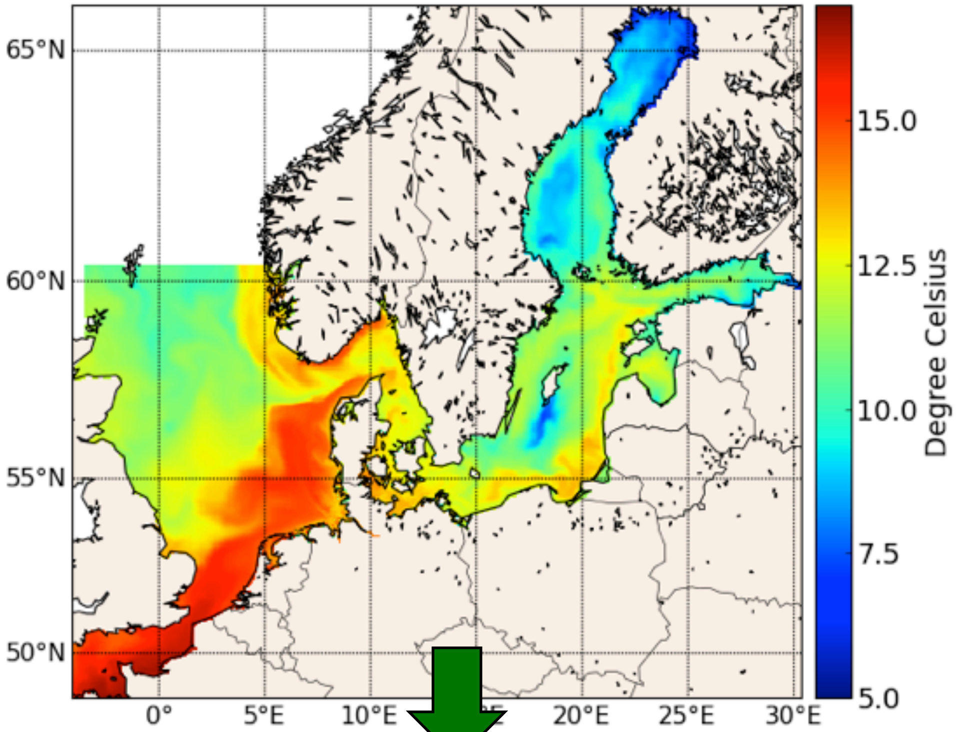
(See Short Course SC1.2 on Friday for methodology)

Application examples

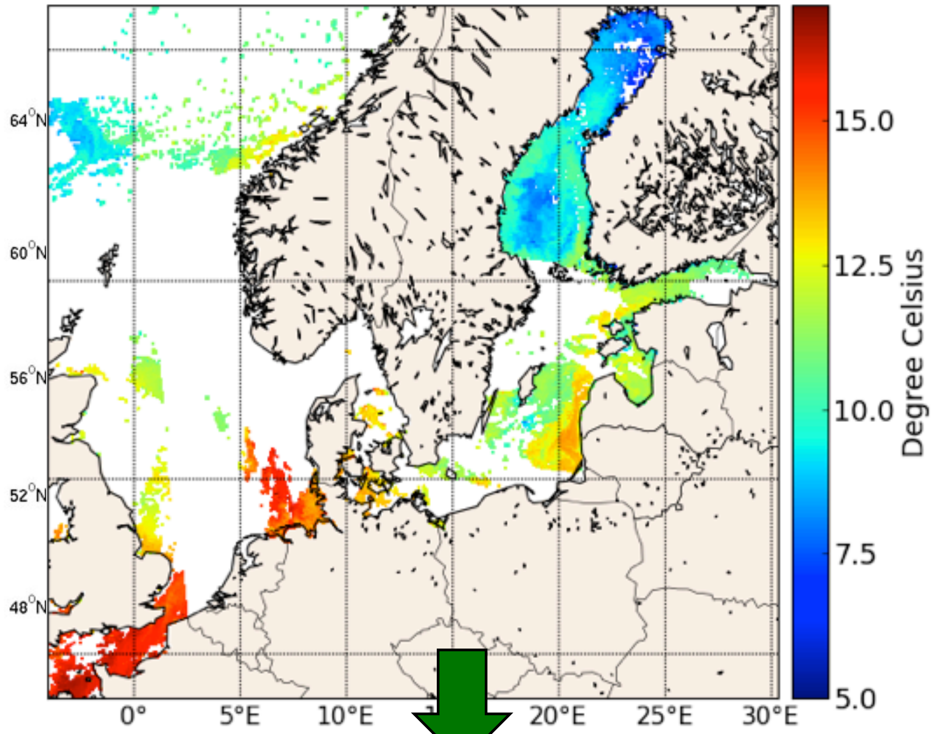
(ocean physics and ocean-biogeochemistry)

Motivation

Model surface temperature



Satellite surface temperature



Combine both sources of information

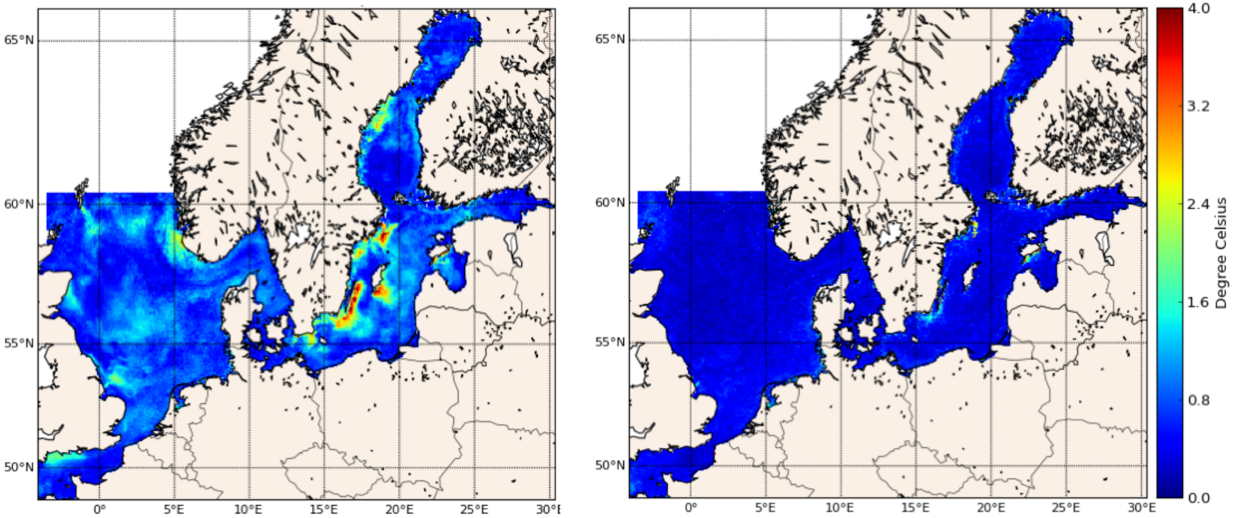
quantitatively by computer algorithm

→ Data Assimilation

DA – effect on Temperature (September 2012)

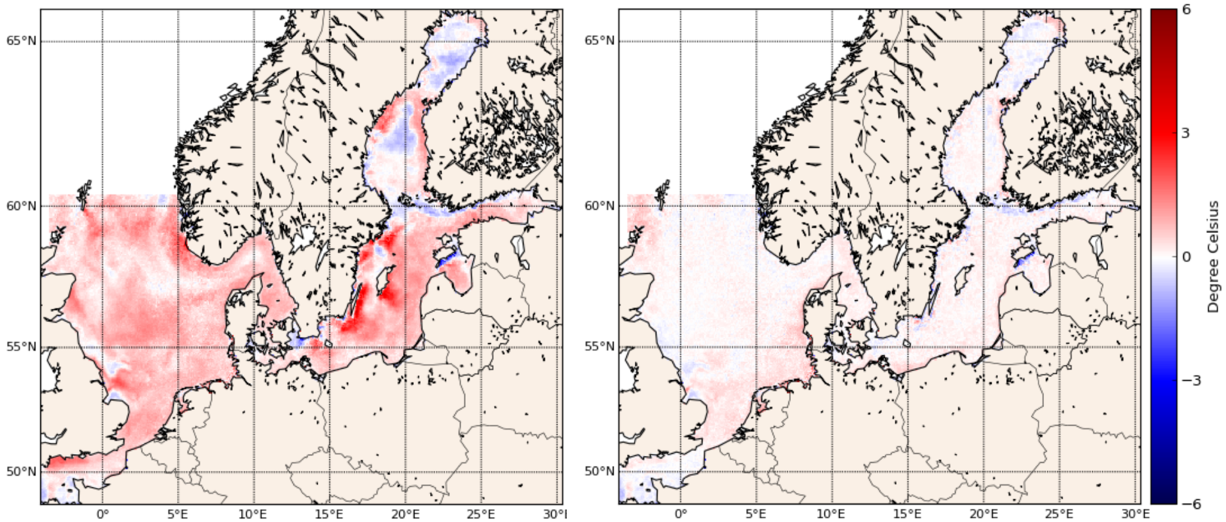
RMS (root-mean-square) deviation

Free run | Assimilation (analysis)



Mean deviation (observation – model)

Free run | Assimilation (analysis)



Assimilate surface temperature each 12 h

Compare assimilated estimate with assimilated surface temperature data (monthly average)

Reduce RMS deviation and mean deviation (bias)

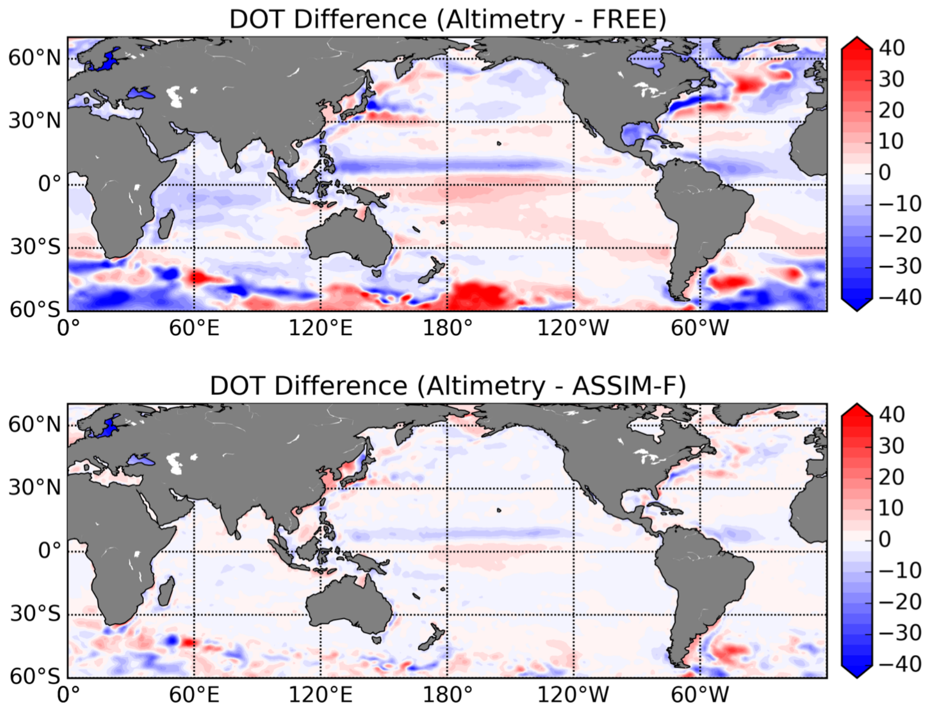
➜ necessary effect

PDAF Parallel Data Assimilation Framework

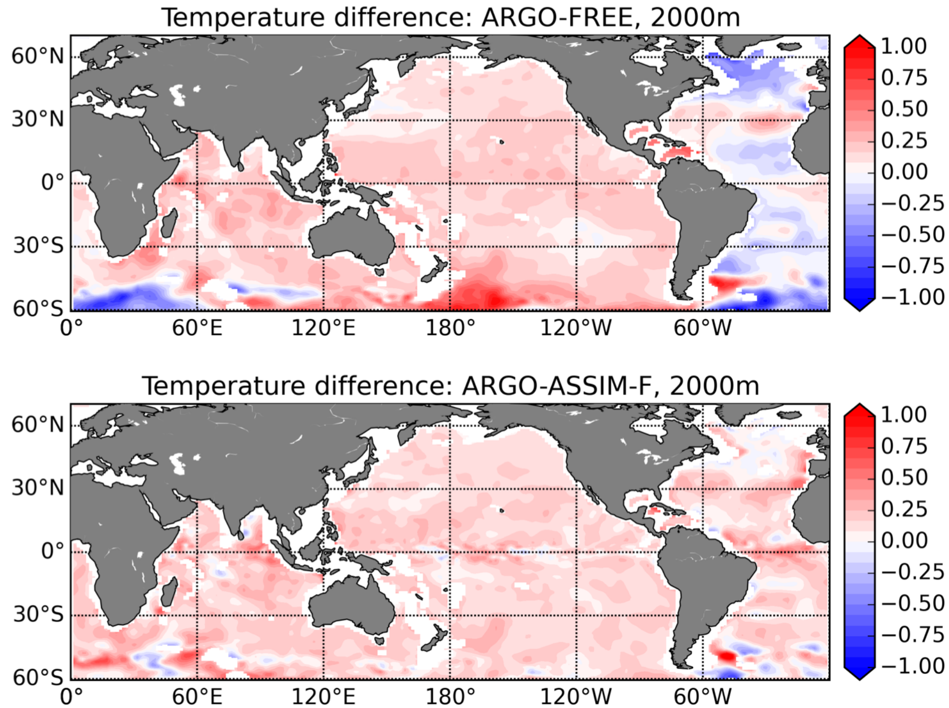
Longe-range effect

Example: Assimilate satellite sea surface height data (DOT)

Reduce difference to assimilated data (necessary)



Improve also temperature at 2000m depth

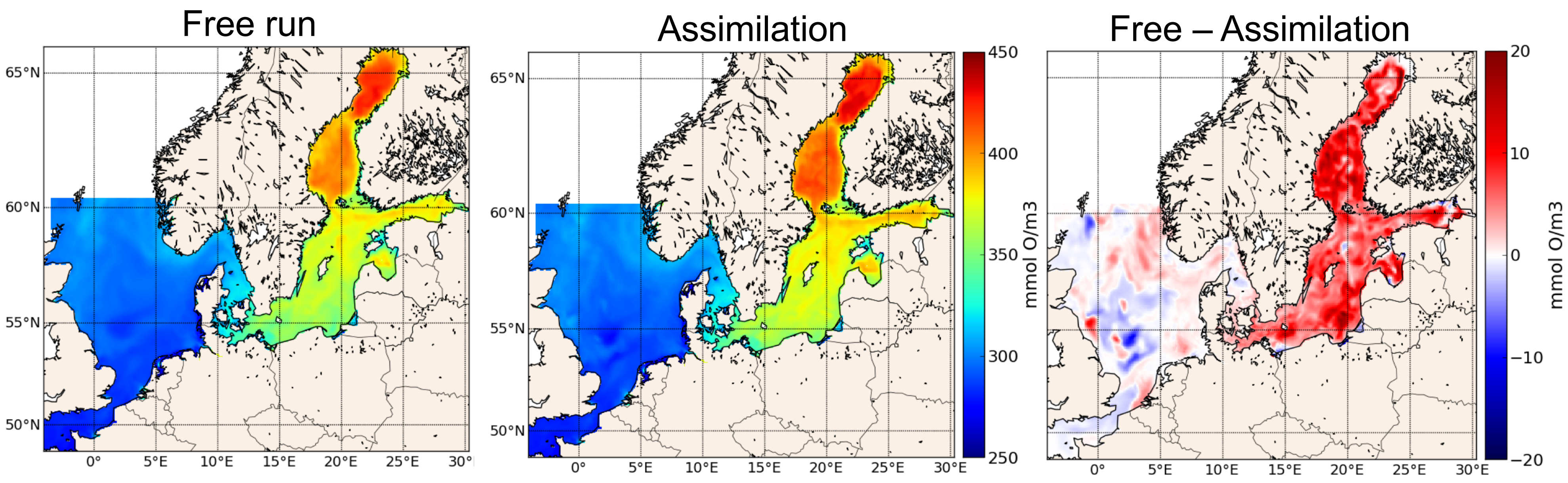


Androsov et al., J. Geodesy, (2019) 93:141–157

PDAF Parallel Data Assimilation Framework

Biogeochemistry: Coupled data assimilation effect

Surface oxygen mean for May 2012 (as mmol O / m^3)



Coupled data assimilation case: physics and biogeochemistry

- Assimilate satellite sea surface temperature observations
- Assimilation directly changes Oxygen and other biogeochemical variables (strongly-coupled assimilation)

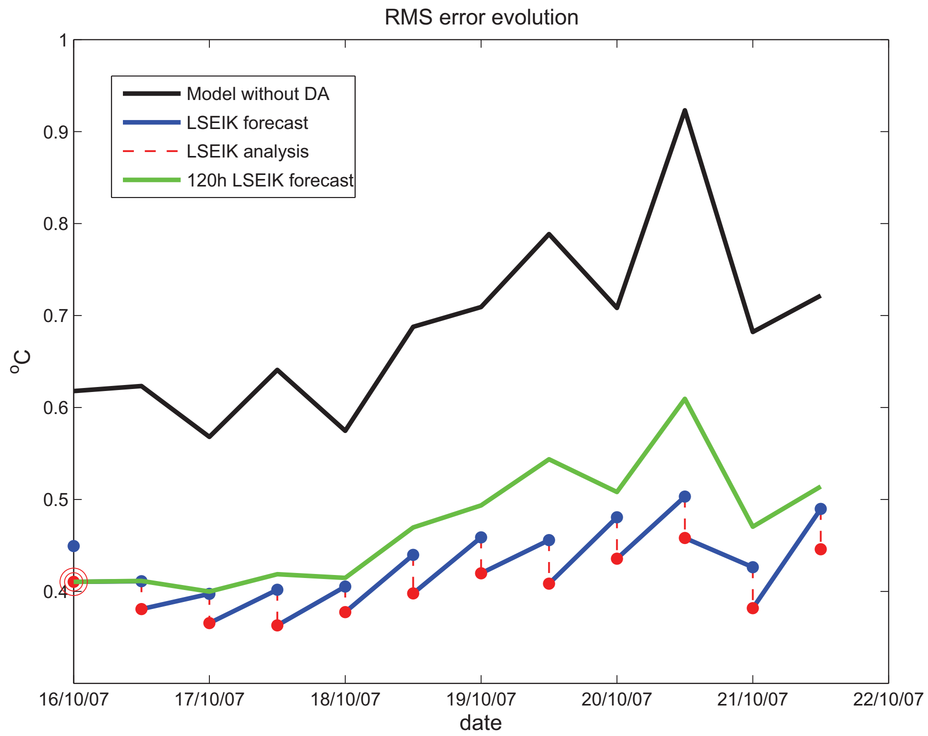
PDAF Parallel Data Assimilation Framework

Improving forecasts

DeMarine

Impact of Assimilation for temperature forecasts

(North & Baltic Seas)



- Very stable 5-days forecasts
- At some point the improvement might break down due to dynamics

S. Losa et al., J. Mar. Syst. 105–108 (2012) 152–162

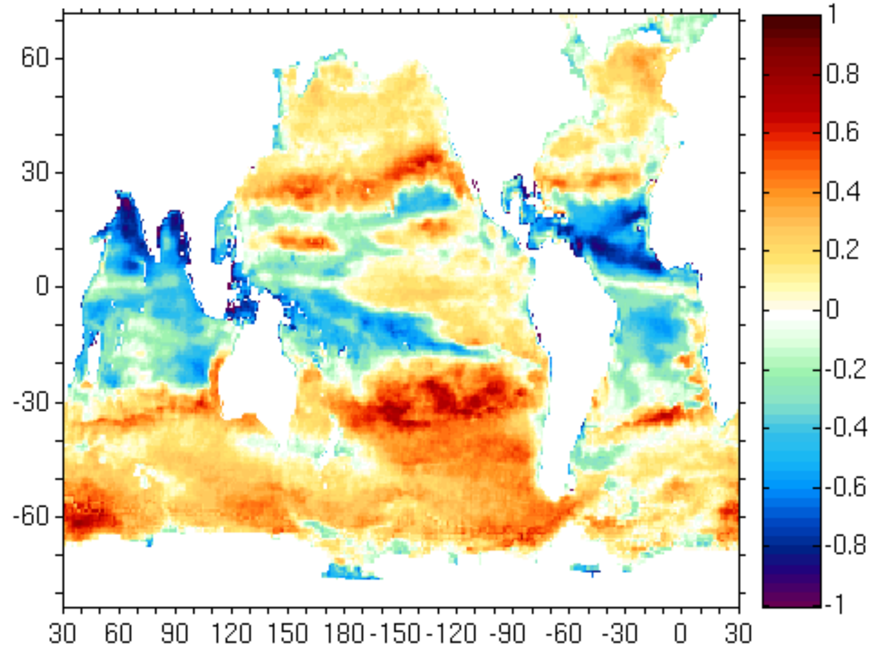
PDAF Parallel Data Assimilation Framework

Bias Estimation

Example: Chlorophyll bias of a biogeochemical model

- *un-biased system*:
  random fluctuation around true state
- *biased system:*
  systematic over- and underestimation
  (common situation with real data)
- *Bias estimation*:
  Separate random from systematic deviations

Logarithmic bias estimate
April 15, 2004

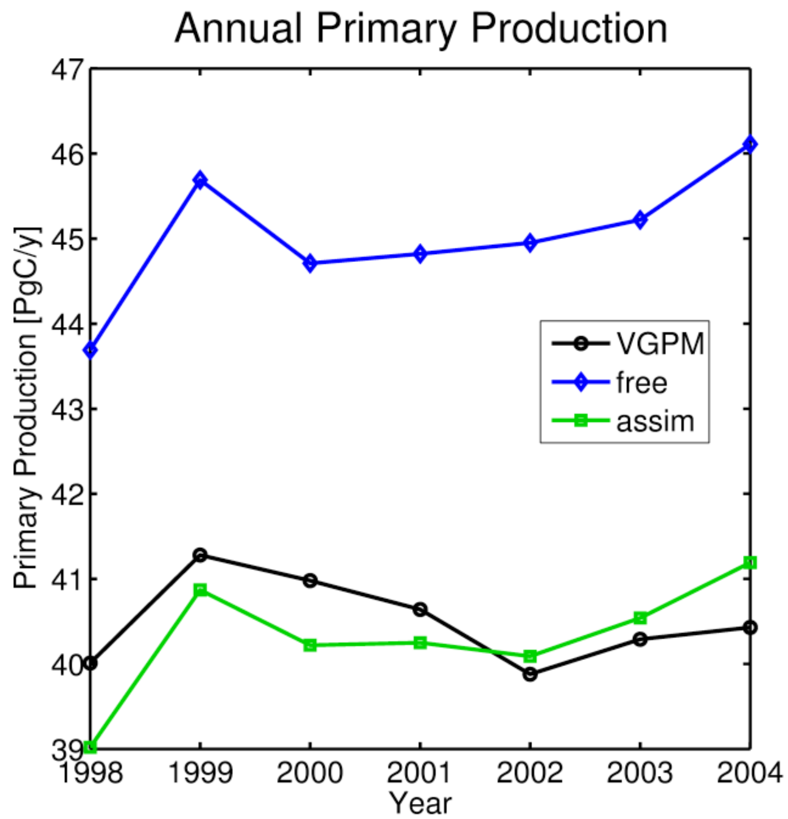


Nerger, L., and W.W. Gregg. J. Marine Systems, 73 (2008) 87-102

Parallel
Data
Assimilation
Framework

Estimate a flux (Primary Production)

- *Primary production is a flux*: Uptake of carbon by phytoplankton
- Model: computed as depth-integrated product of growth-rate times Carbon-to-Chlorophyll ratio
- VGPM: Vertical Generalized Production model - satellite data only
- Primary production from assimilation consistent with VGPM-estimate
- *Important:* Concentration change by assimilation is not primary production



Mean relative difference to VGPM:

Free: 11.2%

Assimilation: -0.5%

(VGPM: Behrenfeld, M.J., P.G. Falkowski., Limnol. Oce. 42 (1997) 1-20)

L. Nerger & W.W. Gregg, *J. Marine Syst.* 68 (2007) 237-254

PDAF Parallel Data Assimilation Framework

Data Assimilation

Combine Models and Observations

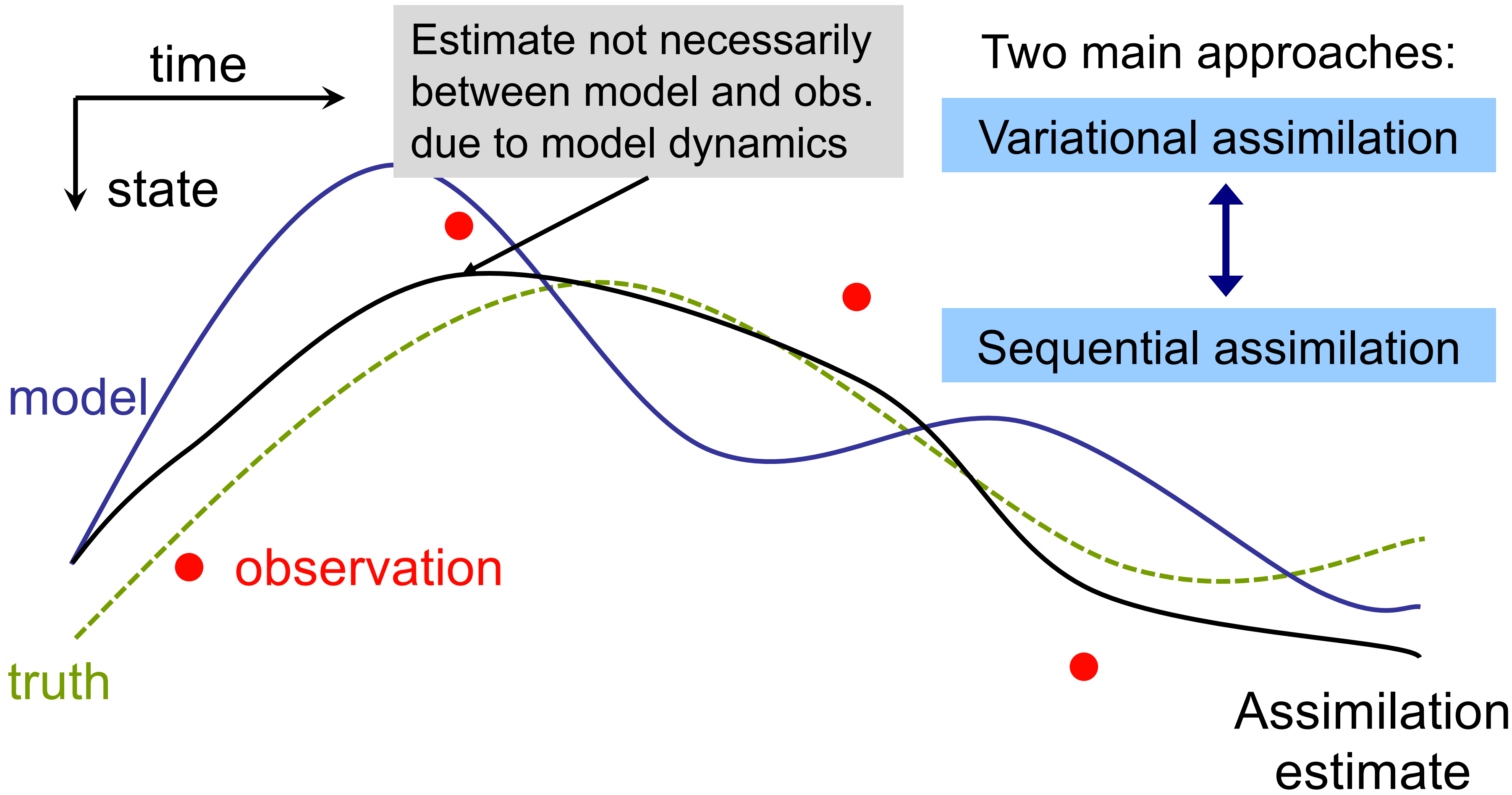
Data Assimilation

Combine model with real data

- Optimal estimation of system state:
  - initial conditions (for weather/ocean forecasts, …)
  - state trajectory (temperature, concentrations, …)
  - parameters (growth of phytoplankton, …)
  - fluxes (heat, primary production, …)
  - boundary conditions and 'forcing' (wind stress, …)
- More advanced: Improvement of model formulation
  - Detect systematic errors (bias)
  - Revise parameterizations based on parameter estimates

Data Assimilation – a general view

Consider some physical system (ocean, atmosphere, land, …)



Optimal estimate basically by least-squares fitting (but constrained by model dynamics)

PDAF Parallel Data Assimilation Framework

Needed for Data assimilation

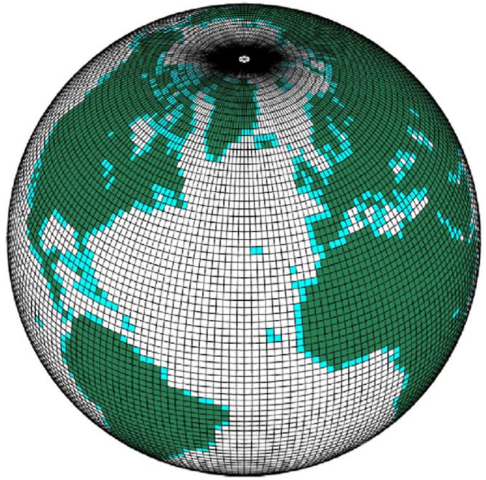
1. Model
   - with some skill
2. Observations
   - with finite errors
   - related to model fields
3. Data assimilation method

Models

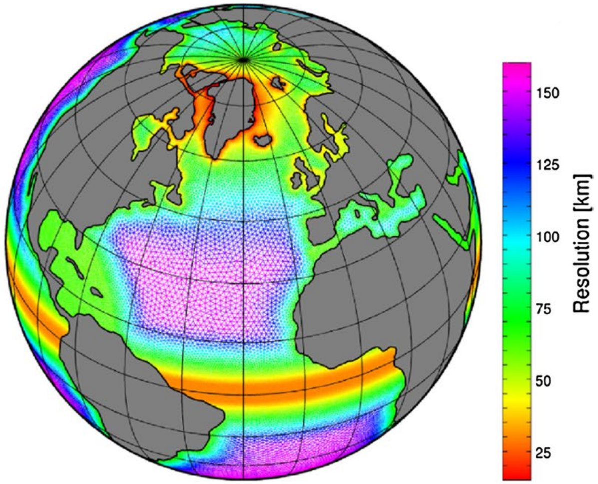
Simulate dynamics of ocean

- Numerical formulation of relevant terms
- Discretization with finite resolution in time and space
- “forced” by external sources (atmosphere, river inflows)

- Uncertainties
  - initial model fields
  - external forcing
  - in predictions due to model formulation



Uniform-resolution mesh



Variable-resolution mesh (ocean model FESOM)

PDAF Parallel Data Assimilation Framework

Observations

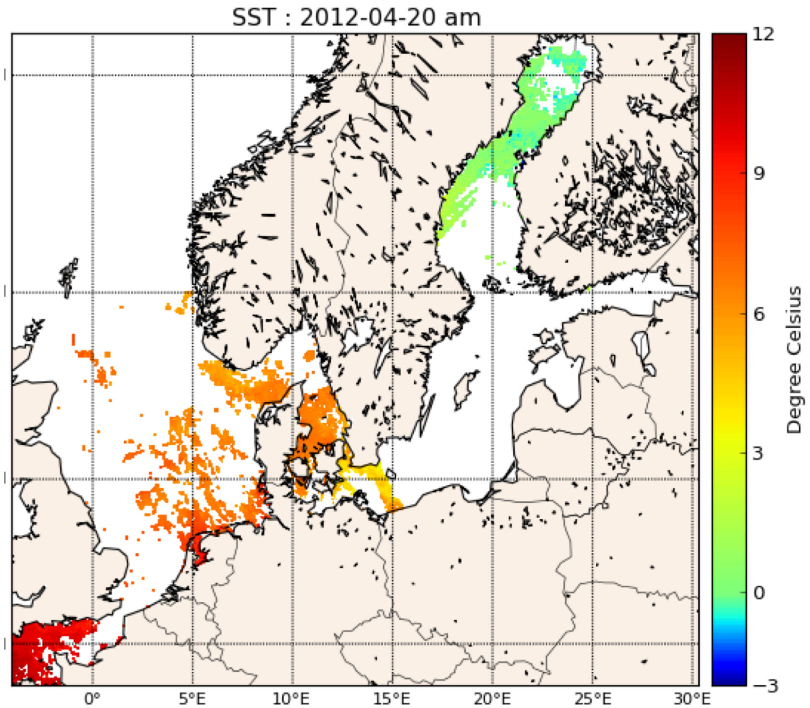
Measure different fields … for example in the Ocean

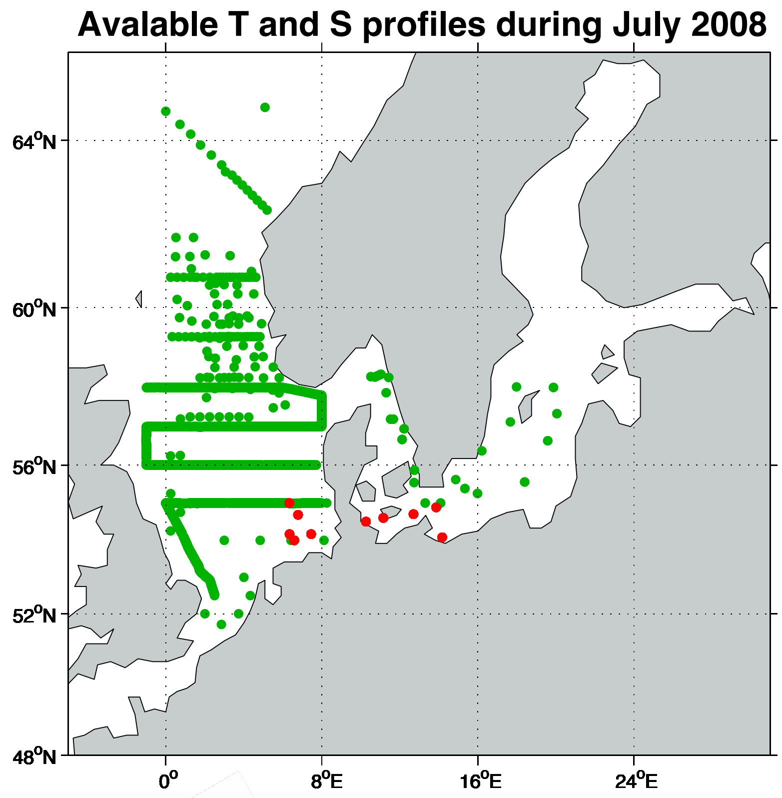
- Remote sensing
  - E.g. surface temperature, salinity, sea surface height, ocean color, sea ice concentrations & thickness
- In situ (ships, autonomous vehicles, …)
  - Argo, CTD, Gliders, …

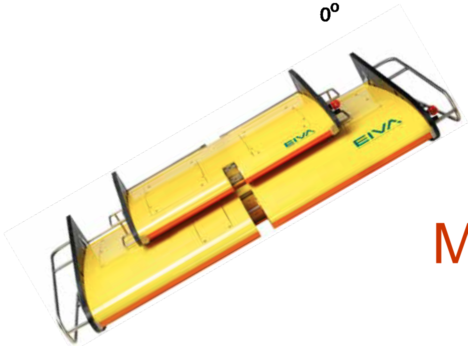
- Data is sparse: some fields, data gaps
- Uncertainties
  - Measurement errors
  - Representation errors:
    Model and data do not represent exactly the same
    (e.g. cause by finite model resolution)

Example: Physical Data in North & Baltic Seas

Satellite surface temperature
(12-hour composite)



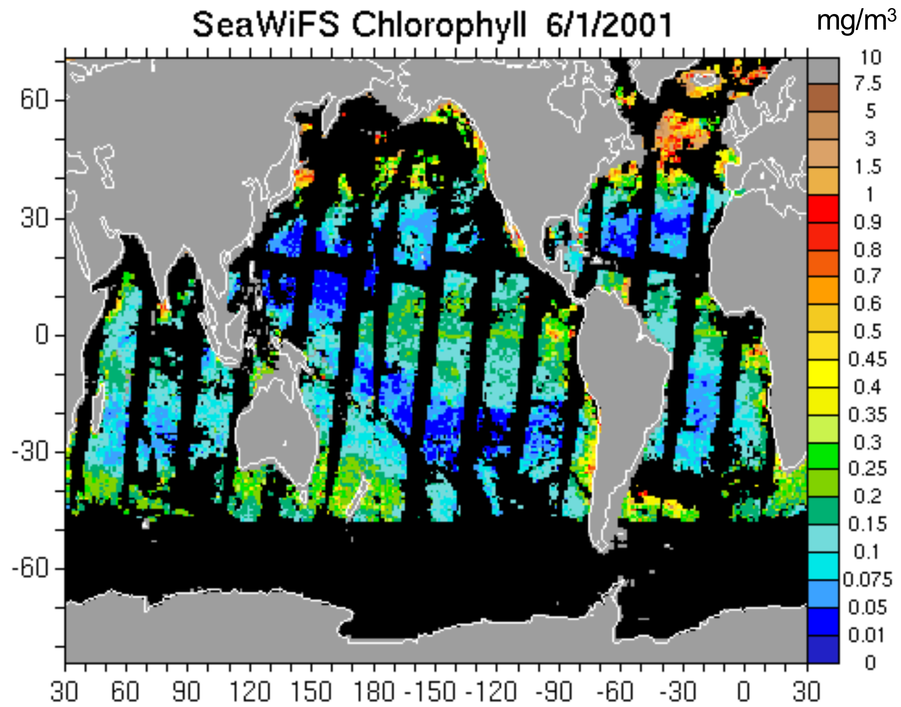


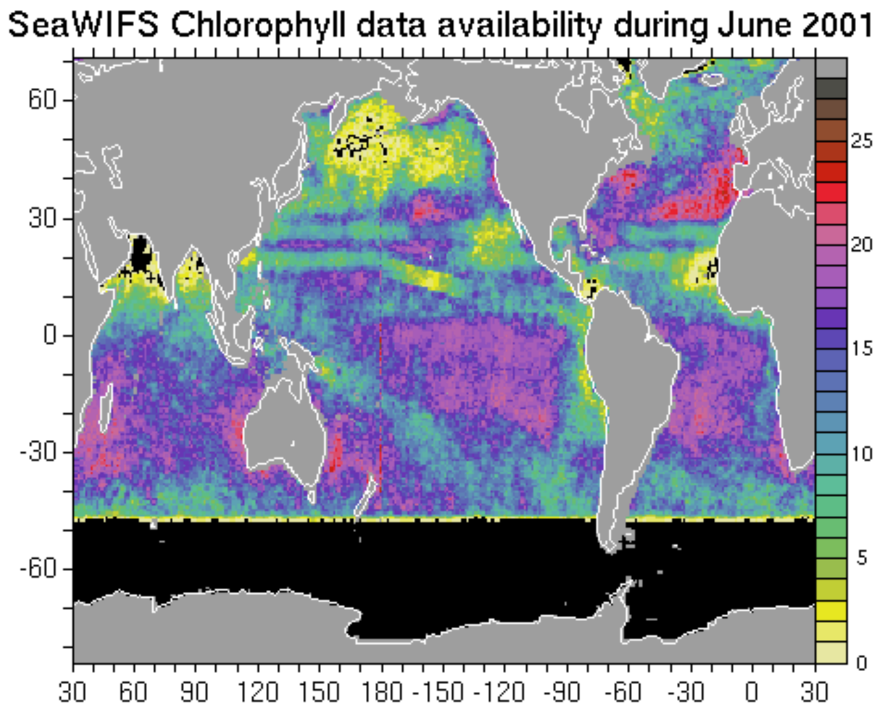


MARNET
stations

Scanfish and
CTD profiles

Example: Chlorophyll-a (SeaWiFS)





Daily gridded SeaWiFS chlorophyll data

- gaps: satellite track, clouds, polar nights
- On model grid: ~13,000-18,000 data points daily (of 41,000 wet grid points)
- irregular data availability

Nerger, L., and W.W. Gregg. J. Marine Systems **68** (2007) 237

PDAF Parallel Data Assimilation Framework

Observation Error Estimates

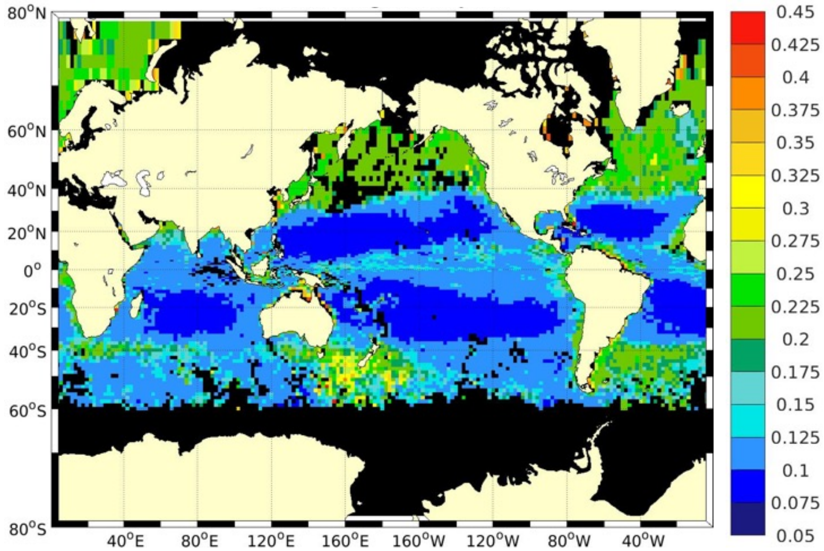
If observation errors available:

- they are typically usable
- usually do not account for representation errors (might be too low)

If no observation errors available:

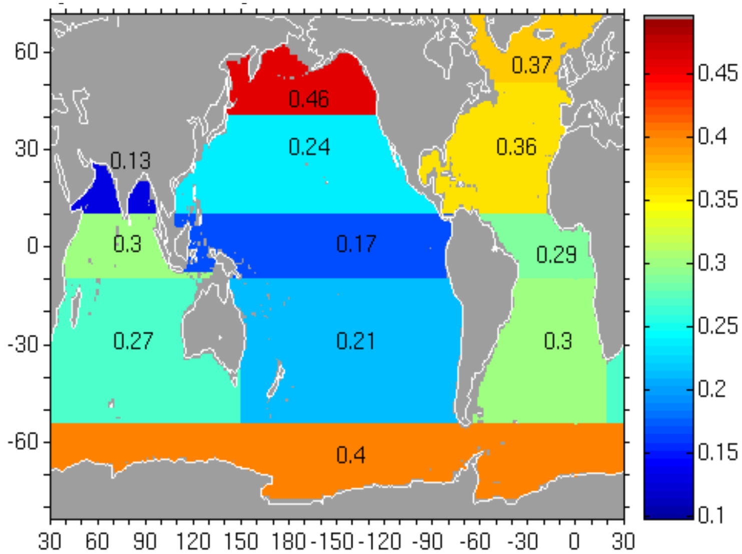
- need to estimate them

logarithmic data errors provided with satellite chlorophyll data (OC-CCI)



Pradhan et al, JGR 2019

data errors from comparison with 2186 collocation points of in situ data (SeaWiFS)



Nerger & Gregg, JMS 2007

Data Assimilation Methods

Combine observations and model state estimate

- Account for uncertainty in observations
- Account for uncertainty in model state estimate
- Account for relations (correlations) between observed part of the model state and unobserved parts

Ensemble Data Assimilation

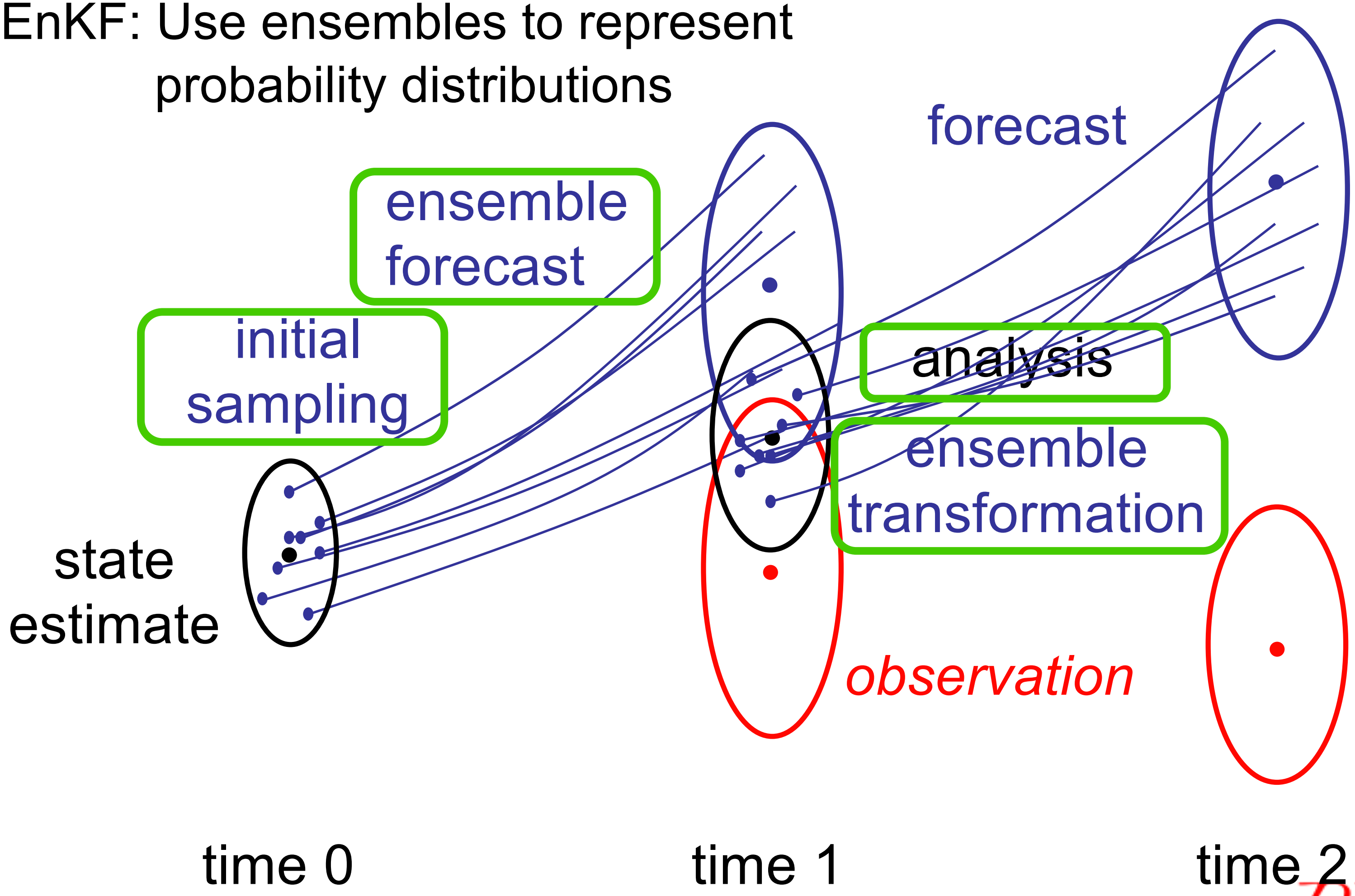
Estimate uncertainty

Ensemble Kalman Filters

First formulated by G. Evensen (EnKF, J. Geophys. Res. 1994)

Kalman filter: express probability distributions by mean and covariance matrix

EnKF: Use ensembles to represent probability distributions



There are many possible choices!

What is optimal is part of our research

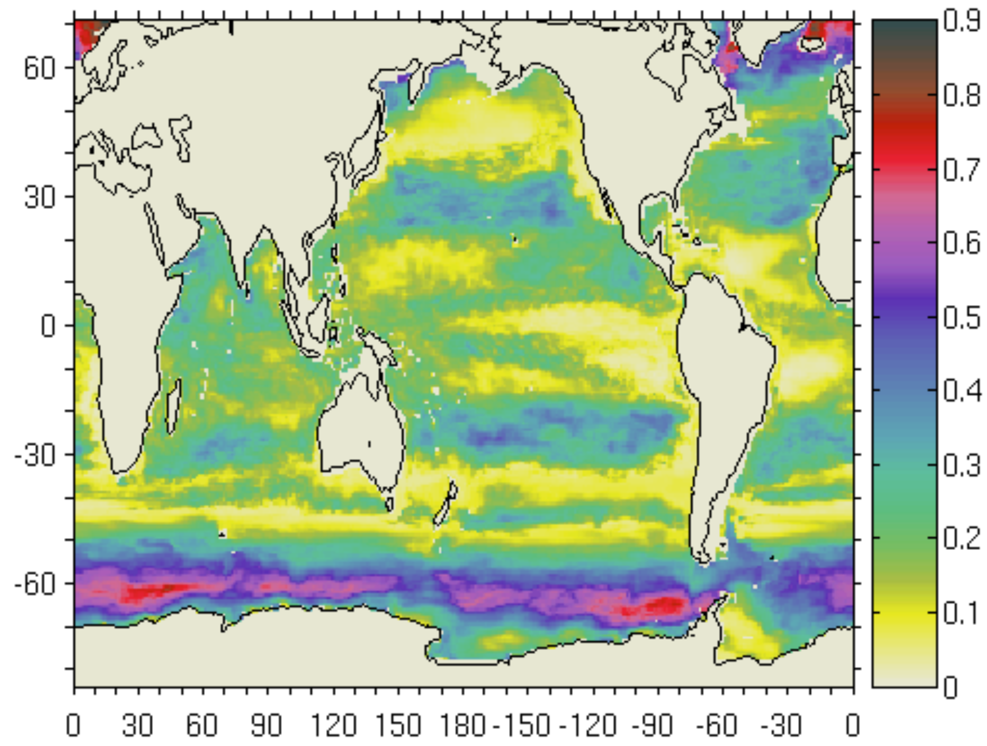
Different choices in PDAF

PDAF Parallel Data Assimilation Framework

Ensemble Covariance Matrix

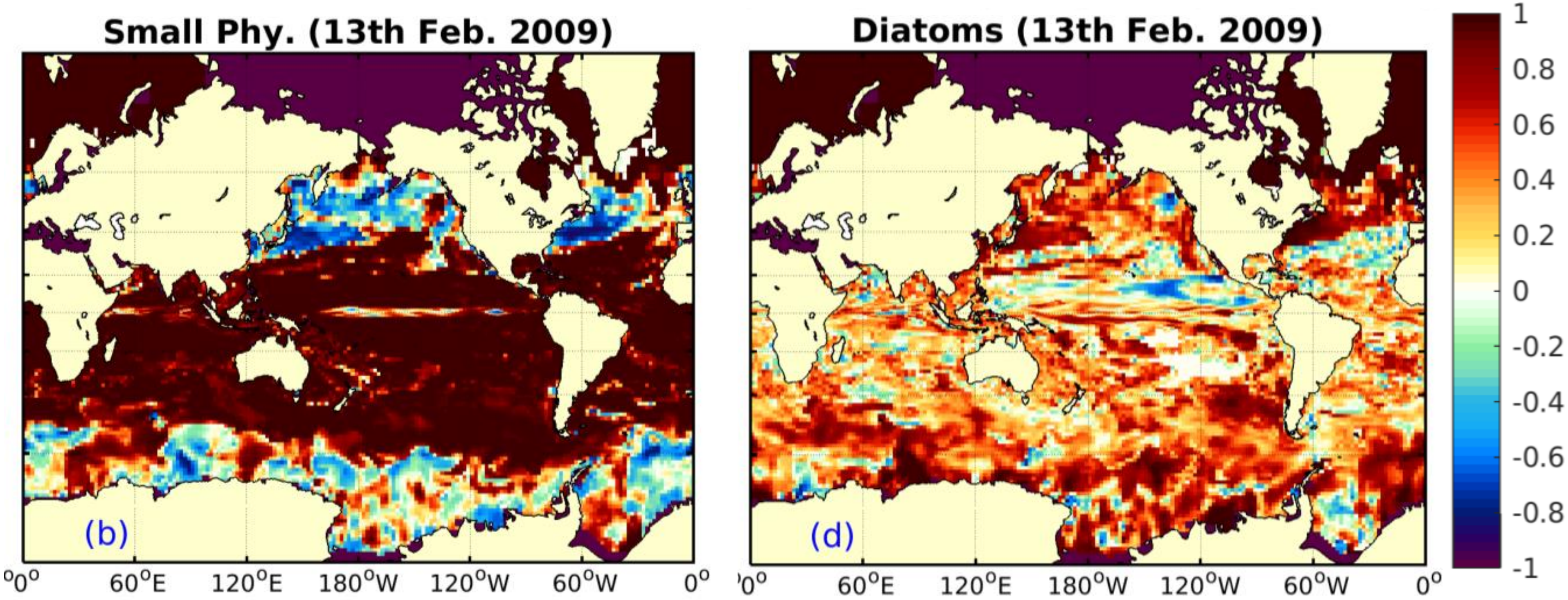
- Provide uncertainty information (variances + covariances)
- Generated dynamically
  by propagating ensemble of model states

Uncertainty: Std. deviation of log Chlorophyll



Ensemble-estimated Cross-correlations

Cross correlations between total chlorophyll and chlorophyll in phytoplankton groups



Cross-correlations are used to correct non-observed quantities from observed ones

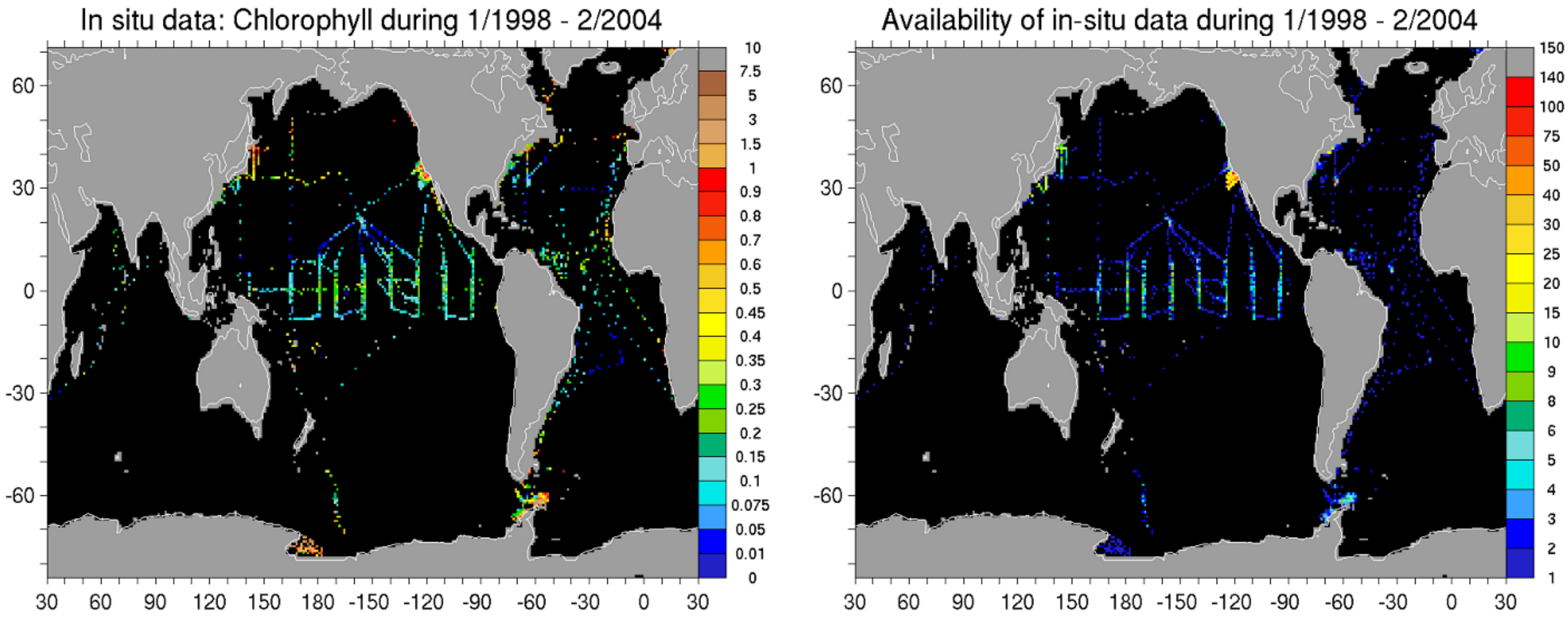
Pradhan et al., J. Geophy. Res. Oceans, 124 (2019) 470-490

Validation of assimilation results

Validating a data assimilation system

- Need independent data for validation
  - Necessary, but not sufficient:
    Reduction of deviation from assimilated data
  - Required:
    - Reduction of deviation from independent data
    - Reduction of errors for unobserved variables
  - Ideally:
    - Reduce error below that of model and data alone
- Want to assimilate all available data (in the ocean)
  - Data-withholding experiments
  - Twin experiments
  - Validate with data of small influence

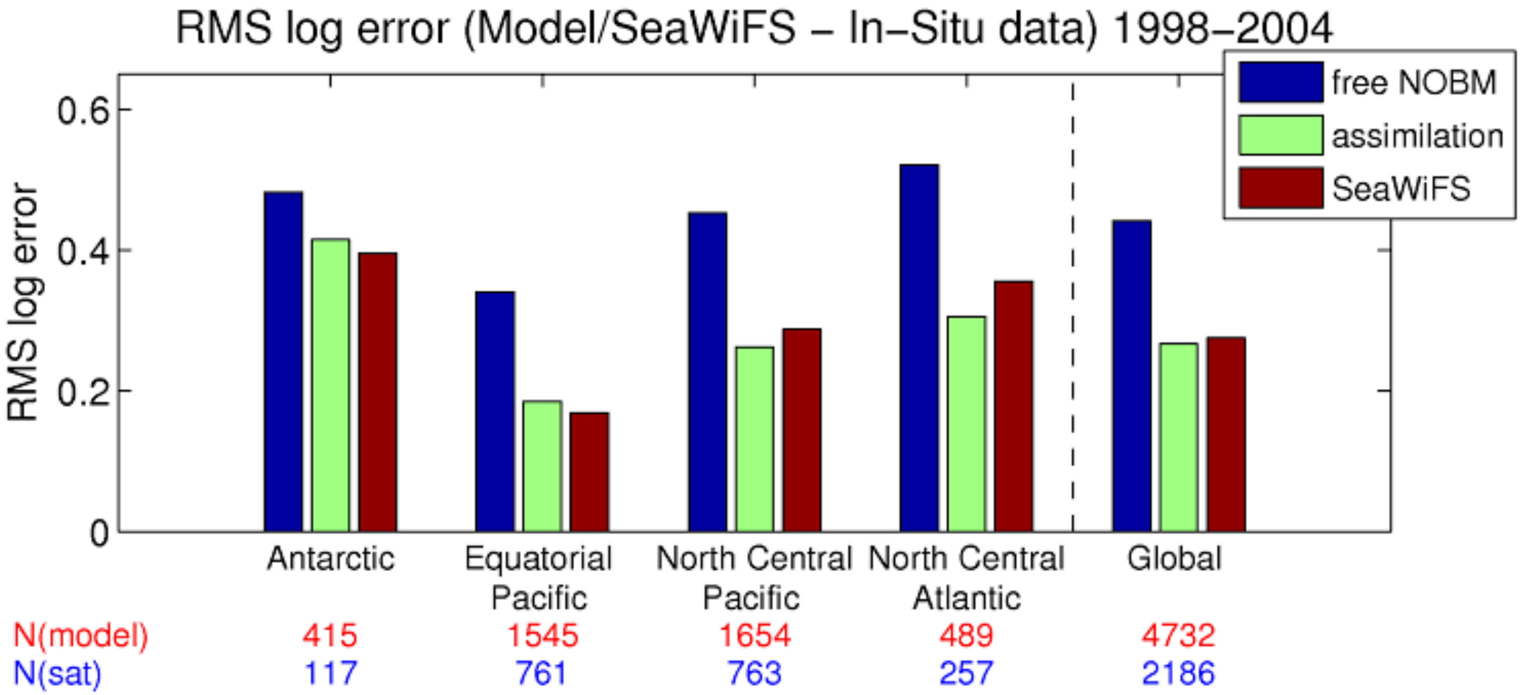
Validation: In-Situ chlorophyll data



- In situ data from SeaBASS/NODC over 1/1998-2/2004
- Independent from SeaWiFS data
  (only used for verification of algorithms)
- North Central Pacific dominated by CalCOFI data
- North Central Atlantic dominated by BATS data

PDAF Parallel Data Assimilation Framework

Comparison with independent data



- Shown basins include about 87% of data
- Compare daily co-located data points

⇒ Assimilation reduces errors significantly

⇒ Error from assimilation lower than SeaWiFS error in many basins and globally

PDAF Parallel Data Assimilation Framework

Quantifying the quality of the assimilation result

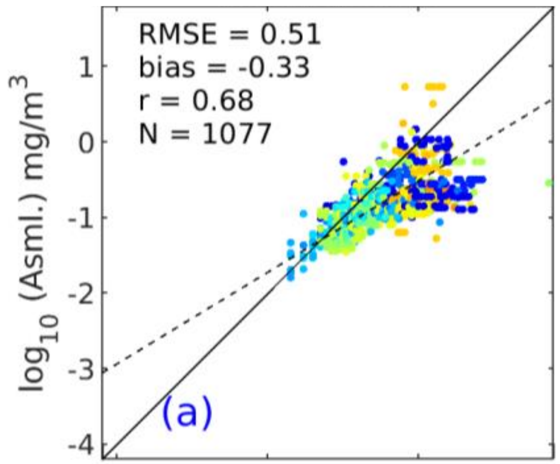
Assess **ensemble mean state**:

Common choices

- RMS (root mean square) errors
- Bias (mean error)
- Correlation

compared to observations

Scatter plot for validation

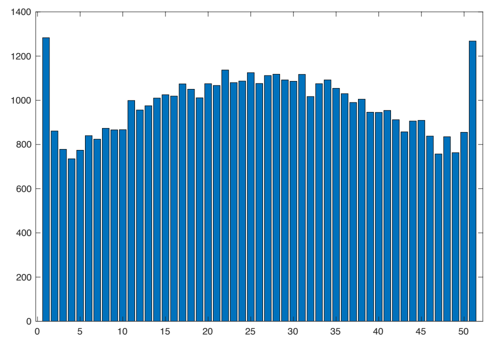


Assess **ensemble quality**:

- Rank histogram
- CRPS (continuous ranked probability score)
- Relative entropy

Particularly relevant when using nonlinear assimilation methods (e.g. particle filters)

Rank histogram, N=50



Essential “Fixes” for Ensemble Filters

Covariance Inflation

Localization

Covariance inflation

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

  → can lead to filter divergence

- Simple remedy

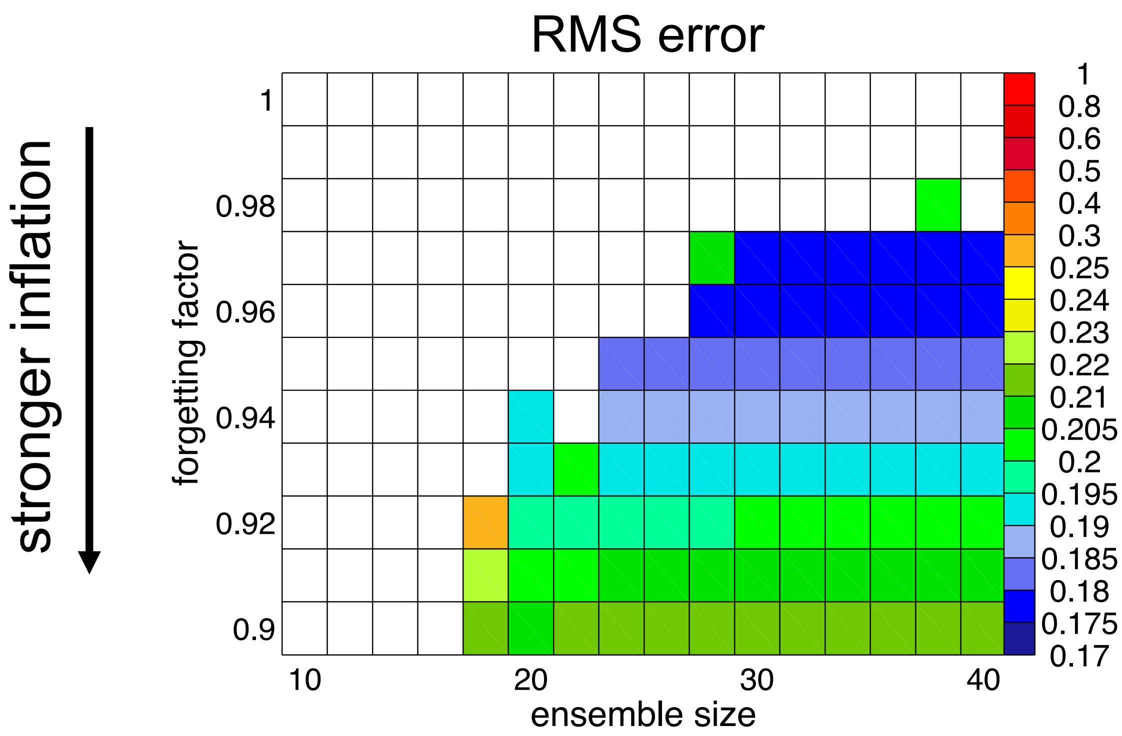
  → Increase error estimate before analysis

- Inflation
  - Increase ensemble spread by constant factor
  - Some filters allow multiplication of a small matrix ("forgetting factor" ≤1; computationally very efficient)
  - Needs to be experimentally tuned

(Mathematically, this is a regularization)

Impact of inflation on stability & performance

Experiments with Lorenz96 model
(available with PDAF)



Lorenz96 model

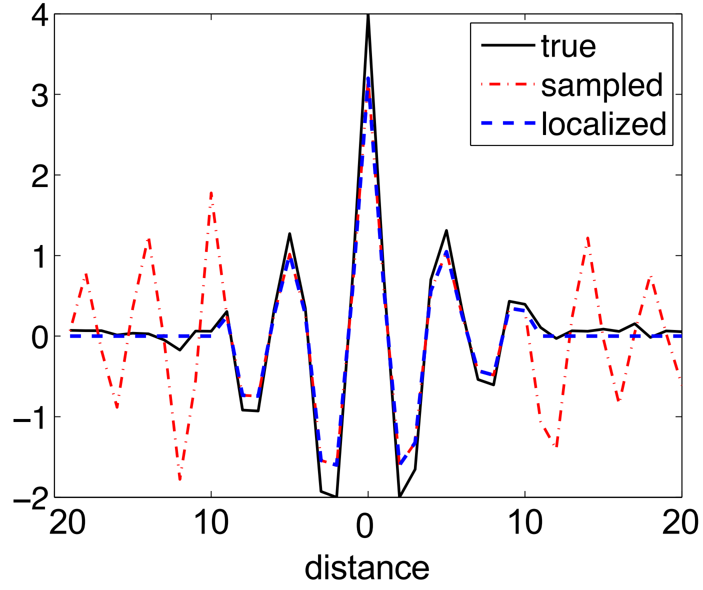
- a widely used toy model
- one-dimensional period wave
- chaotic dynamics
- included in PDAF release

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

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 small covariances!

Example: Sampling error and localization



- Remove estimated long-range correlations

  → Increases degrees of freedom for analysis (globally not locally!)

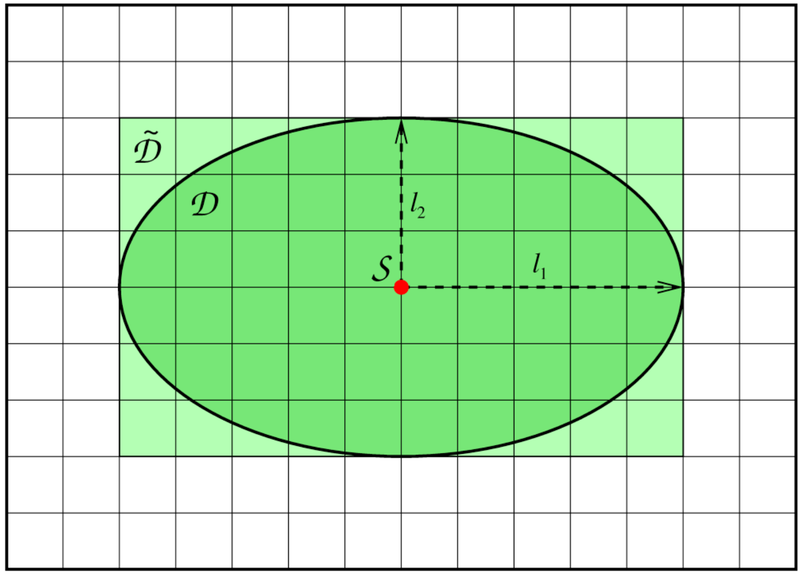
  → Increases size of analysis correction

(introduced for EnKFs by Houtekamer & Mitchell 1998)

Observation Localization

Local Analysis:

- Update small regions (like single vertical columns) allows to define distance
- Use only observations within some distance around this region
- State update and ensemble transformation fully local



S: Analysis region
D: Corresponding data region

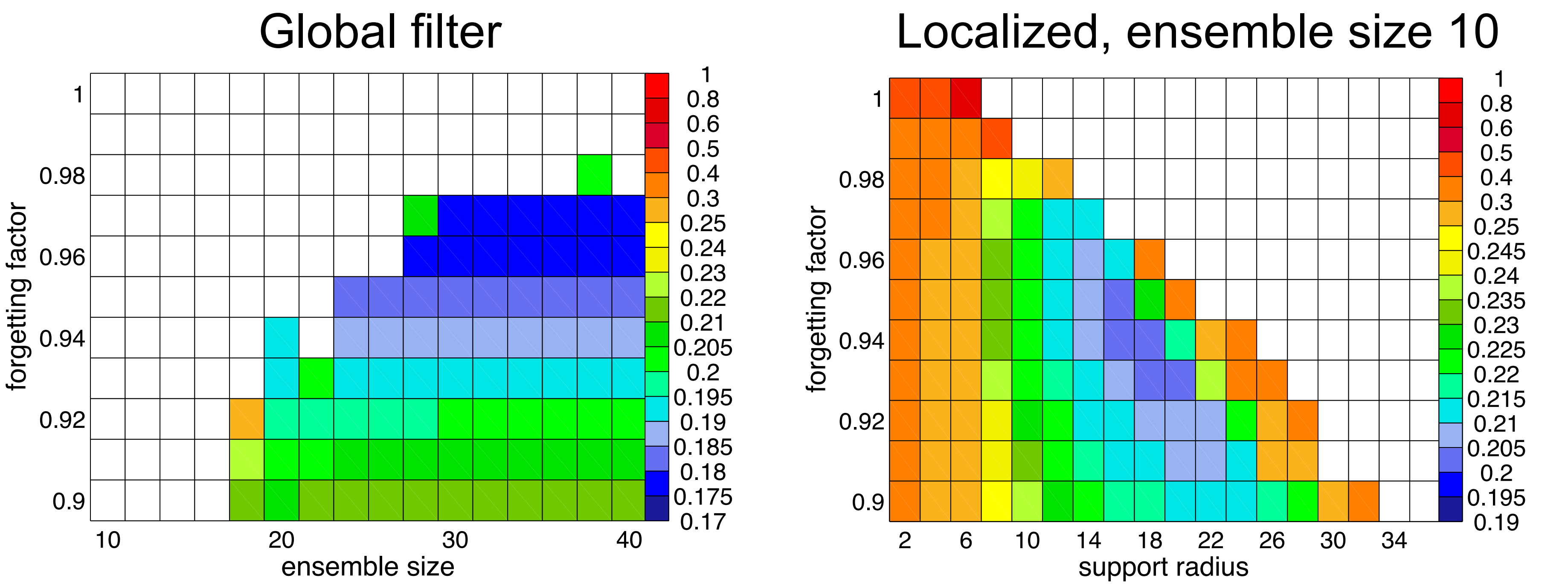
Observation localization:

- Down-weight observations with increasing distance

PDAF Parallel Data Assimilation Framework

Impact of inflation and localization

Experiments with Lorenz96 model



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

Overview

- What can we expect to achieve with data assimilation?
- What do we need for data assimilation?
- How does ensemble data assimilation work?
- How can we apply ensemble data assimilation?

Please note:
We omit equations of assimilation methods because you can apply PDAF without knowing them

(See Short Course SC1.2 on Friday for methodology)

2

Implementation Concept of PDAF

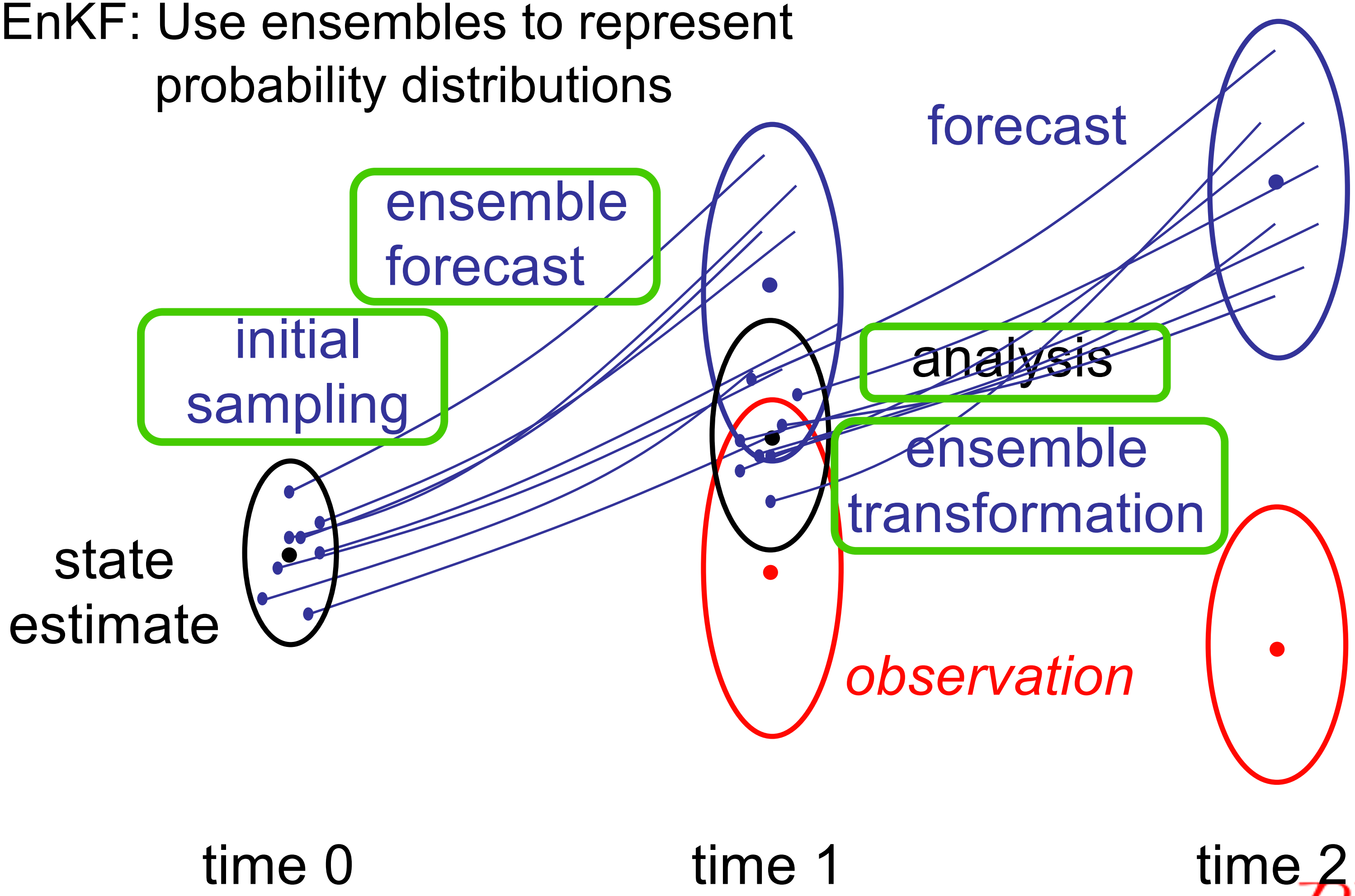
(Parallel Data Assimilation Framework)

Ensemble-based Kalman Filter

First formulated by G. Evensen (EnKF, J. Geophys. Res. 1994)

Kalman filter: express probability distributions by mean and covariance matrix

EnKF: Use ensembles to represent probability distributions



There are many possible choices!

What is optimal is part of our research

Different choices in PDAF

Computational and Practical Issues

- Running a whole model ensemble is costly
- Ensemble propagation is naturally parallel (all independent)
- Ensemble data assimilation methods need tuning
- No need to go into model numerics (just model forecasts)
- Filter step of assimilation only needs to know:
  - Values of model fields an their location
  - Observed values, their location and uncertainty

→ Ensemble data assimilation can be implemented in form of a generic code + case-specific routines

→ Can be used without knowing the exact details of the filter algorithm

PDAF: A tool for data assimilation

PDAF Parallel Data Assimilation Framework

PDAF - Parallel Data Assimilation Framework

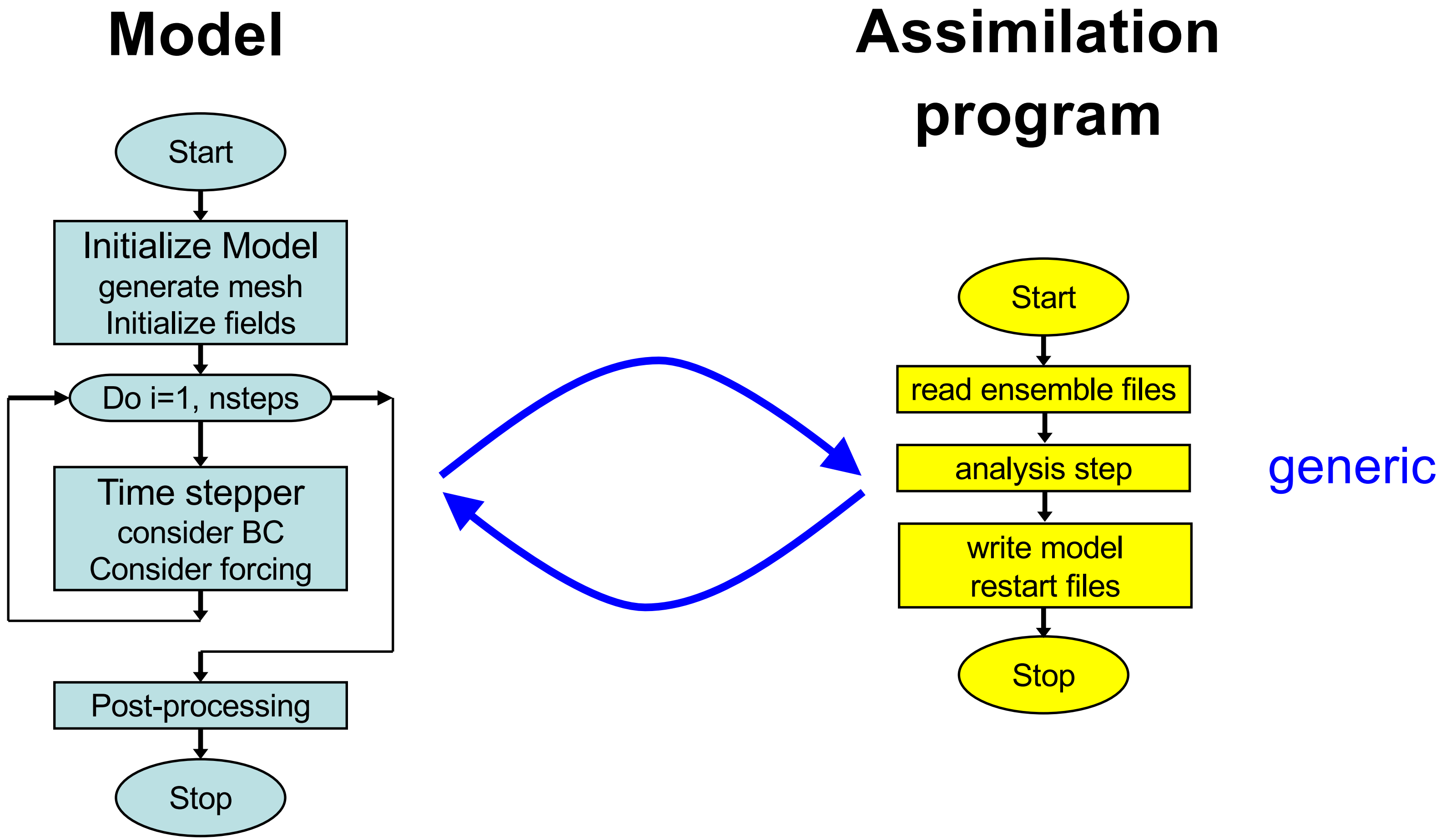
- a program library for ensemble data assimilation
- provide support for parallel ensemble forecasts
- provide fully-implemented & parallelized filters and smoothers (EnKF, LETKF, NETF, EWPF … easy to add more)
- easily useable with (probably) any numerical model (applied with MITgcm, NEMO, FESOM, HBM, TerrSysMP, …)
- run from laptops to supercomputers (Fortran, MPI & OpenMP)
- first public release in 2004; continued development
- ~350 registered users; community contributions

Open source:
Code and documentation available at

http://pdaf.awi.de

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

Offline coupling – separate programs



For each ensemble state

- Initialize from restart files
- Integrate
- Write restart files

- Read restart files (ensemble)
- Compute analysis step
- Write new restart files

PDAF Parallel Data Assimilation Framework

Offline coupling - Efficiency

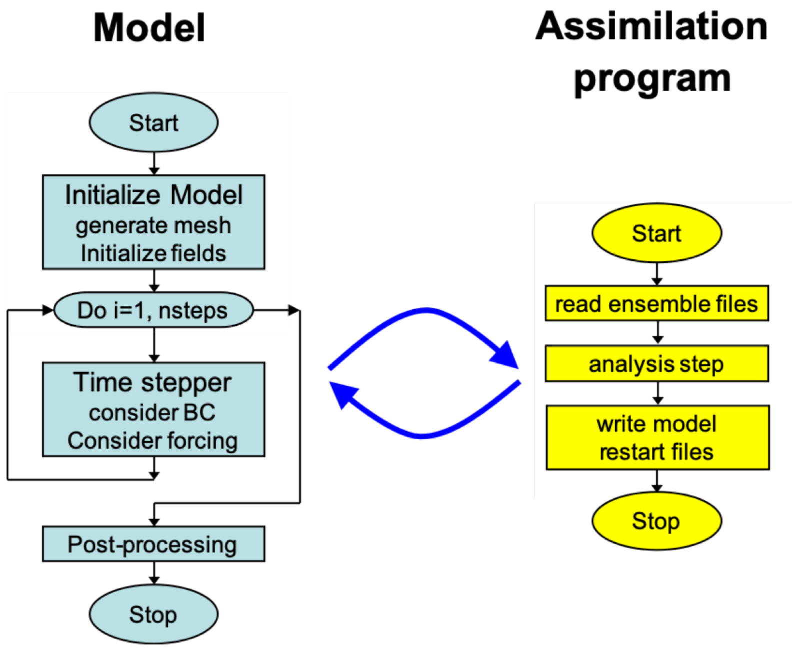
Offline-coupling is simple to implement
but can be very inefficent

Example:
Timing from atmosphere-ocean
coupled model (AWI-CM)
with daily analysis step:

Model startup:	95 s	overhead
Integrate 1 day:	28 s	
Model postprocessing:	14 s	
Analysis step:	1 s	

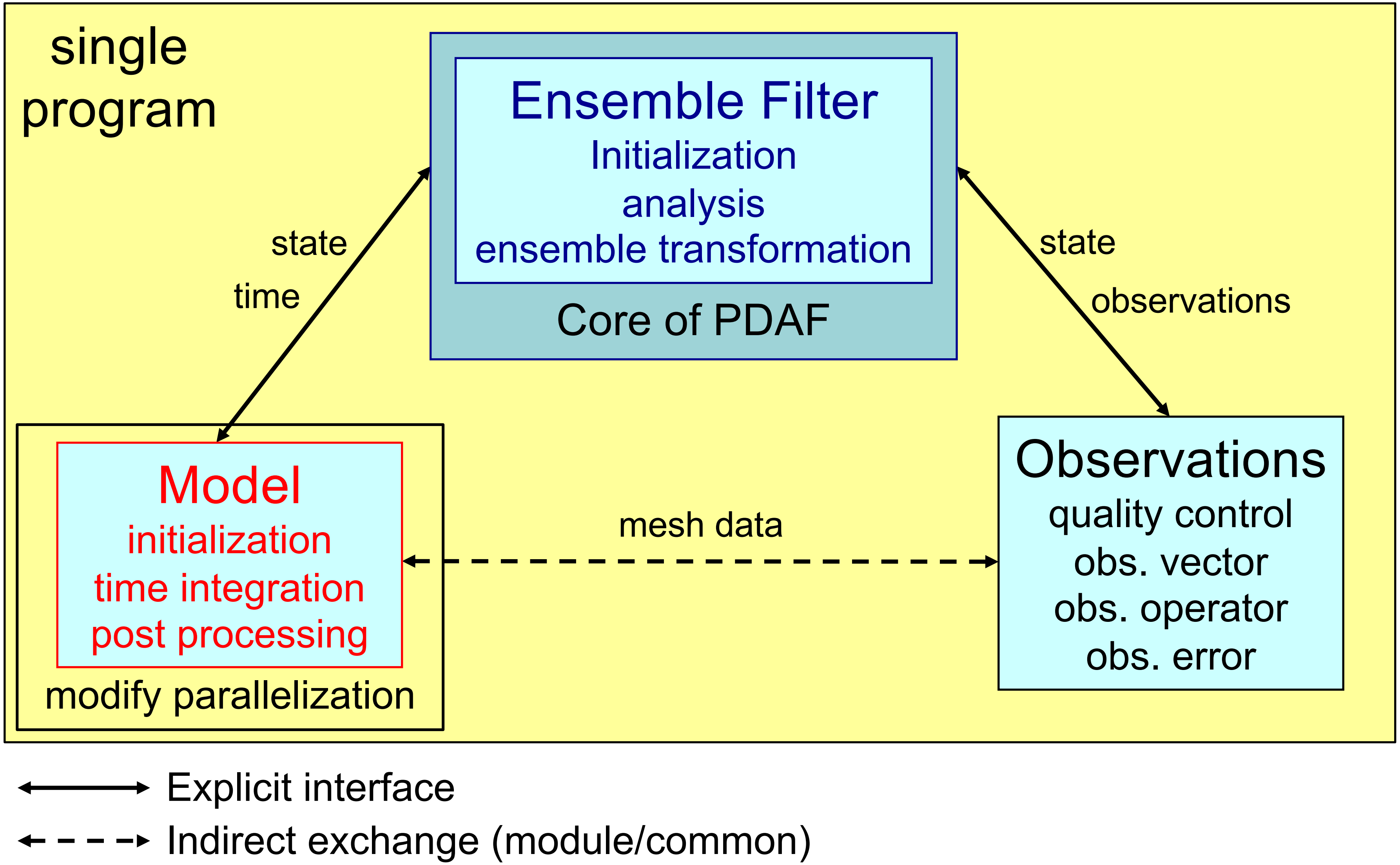
Restarting this model is ~3.5 times
more expensive than integrating 1 day

➜ avoid this for data assimilation



PDAF Parallel Data Assimilation Framework

Components of an Assimilation System

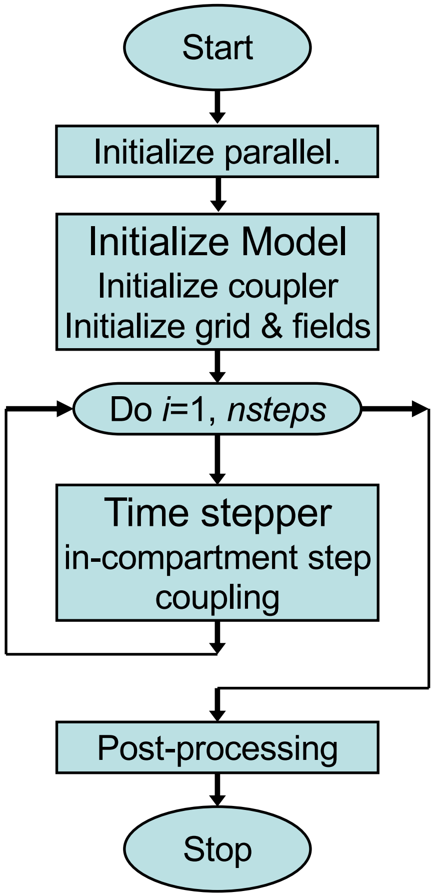


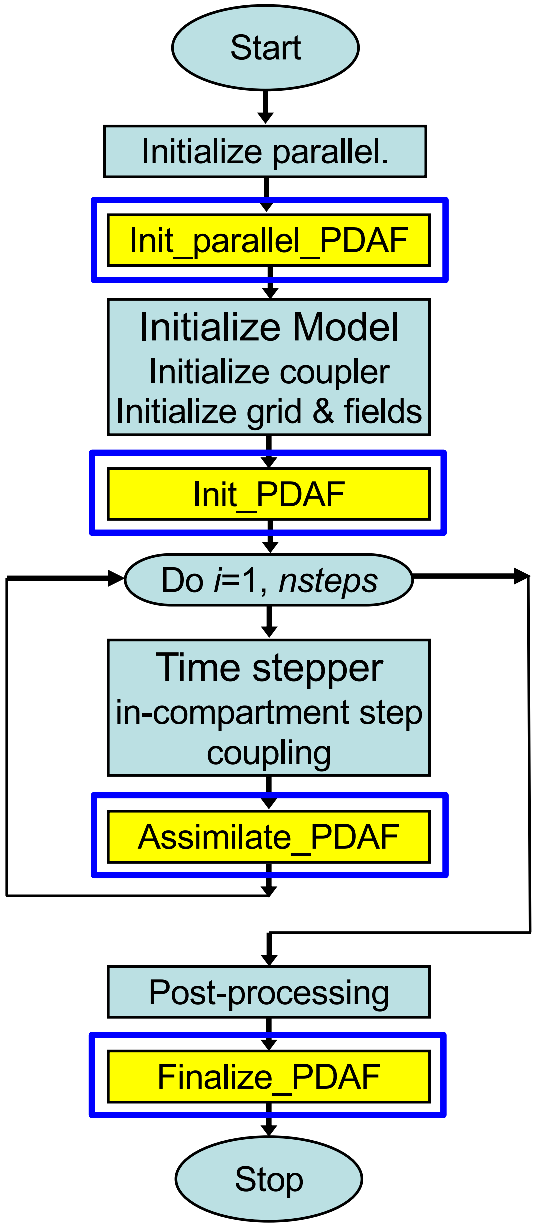
Extending a Model for Data Assimilation

Model

single or multiple executables

coupler might be separate program





Extension for data assimilation

plus:
Possible model-specific adaption

for MITgcm:
adapt name of STDOUT files for ensemble

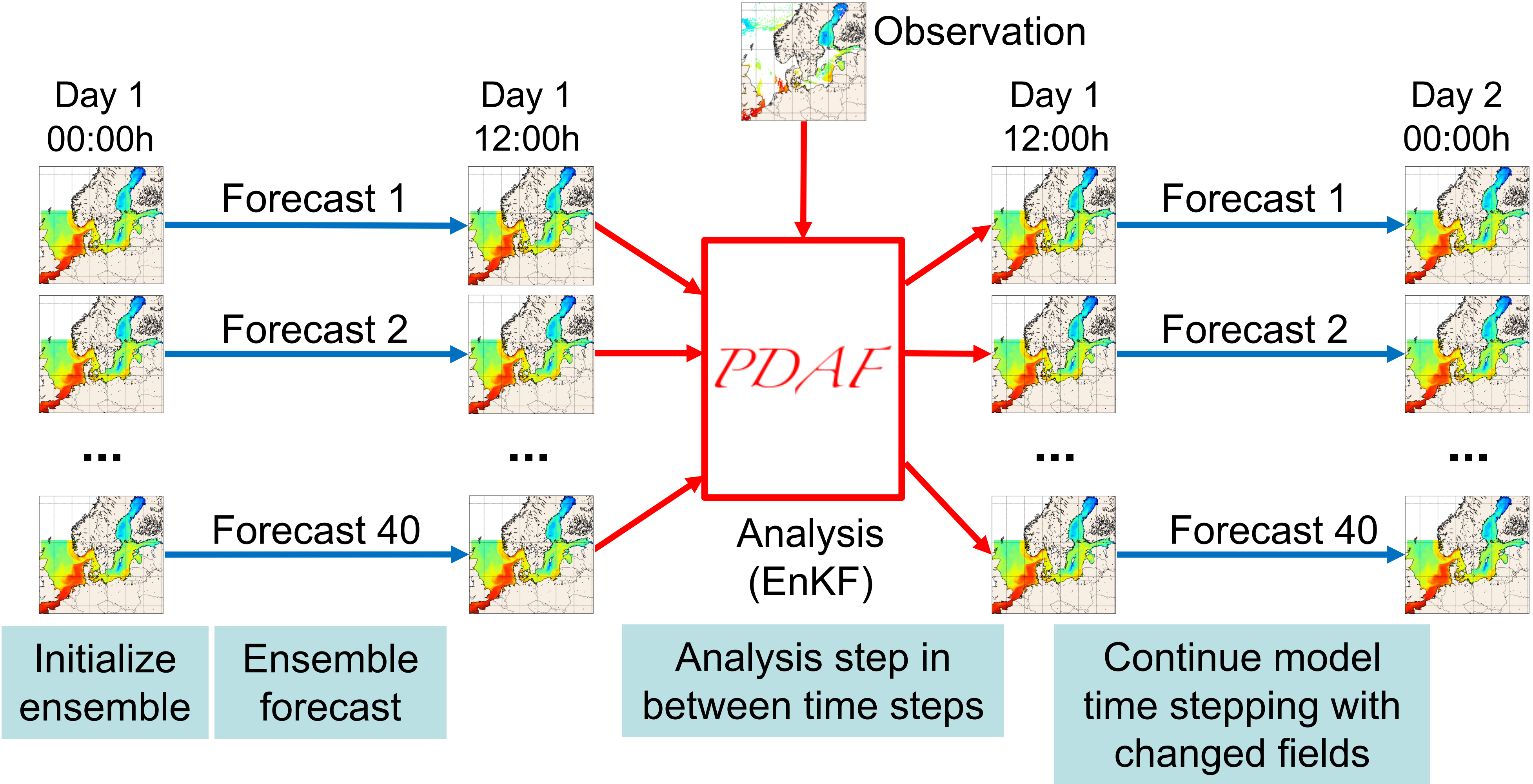
revised parallelization enables ensemble forecast

PDAF Parallel Data Assimilation Framework

Augmenting a Model for Data Assimilation

Couple PDAF (Parallel Data Assimilation Framework) with model

- Modify model to simulate ensemble of model states
- Insert correction step (analysis) to be executed each 12 model hours
- Run model as usual, but with more processors and additional options



PDAF model binding routines

Interface routines

- `init_parallel_pdaf`, `init_pdaf`, `assimilate_pdaf`, `finalize_pdaf`

Call-back routines

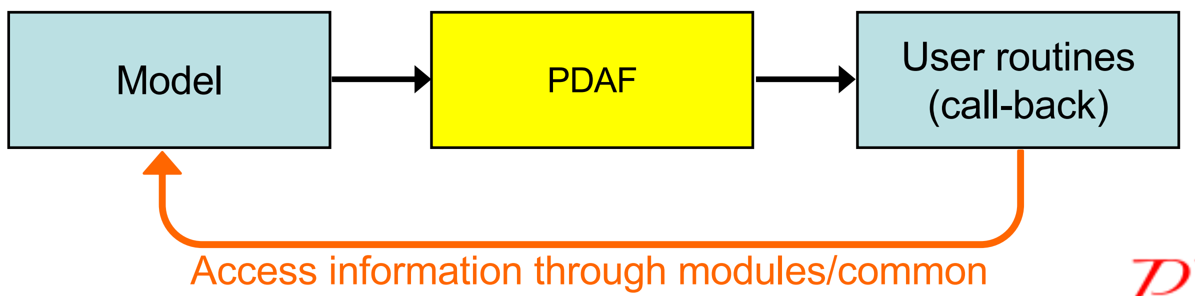
- Set number of time steps between analysis steps
- Write model fields into PDAF's state vector and back into model fields
- Observation handling

PDAF release includes set of model binding routines for MITgcm

- ➢ for a simple test case
- ➢ just download and adapt for your needs
- ➢ (NEMO will be next)

PDAF interface structure

- Interface routines call PDAF-core routines
- PDAF-core routines call case-specific routines provided by user (included in model binding set)
- User-supplied call-back routines for elementary operations:
  - field transformations between model and filter
  - observation-related operations
- User supplied routines can be implemented as routines of the model (for MITgcm: Fortran-77 fixed-form source code)



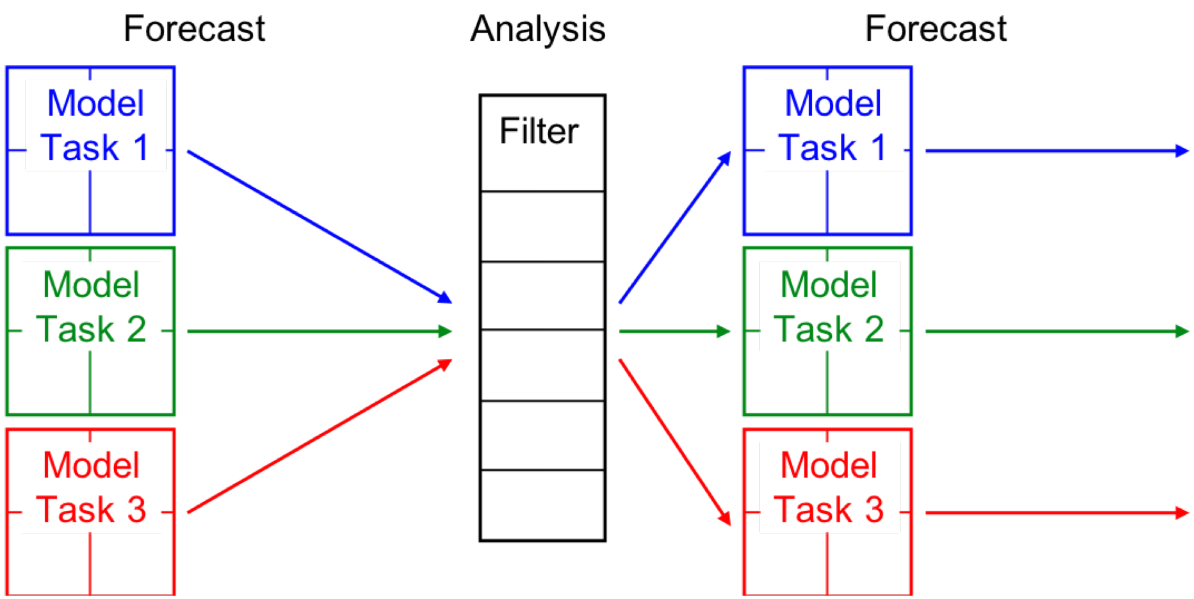
Init_parallel_PDAF

Parallelization of Assimilation Program

We use MPI (Message Passing Interface)

- standard for highly scaling parallelization
- used by most large-scale models

Only need to do this once for a model (e.g. done for MITgcm)



Init_PDAF

Initialization of Assimilation

Set parameters, for example

- select filter
- set ensemble size

Calls `PDAF_init`

- initialization routine of framework
- provide parameters according to interface
- provide MPI communicators
- provide name of routine for ensemble initialization

Ensemble initialization routine – called by PDAF_init

- a “call-back routine”
- defined interface: provides ensemble array for initialization
- user-defined initialization

Simple Subroutine Interfaces

Example: ensemble initialization

```
SUBROUTINE init_ens_pdaf(filtertype, dim, dim_ens, state,
matrU, ens, flag)

  IMPLICIT NONE

! ARGUMENTS:
  INTEGER, INTENT(in) :: filtertype ! Type of filter
  INTEGER, INTENT(in) :: dim        ! Size of state vector
  INTEGER, INTENT(in) :: dim_ens    ! Size of ensemble
  REAL, INTENT(out) :: ens(dim, dim_ens) ! state ensemble
  INTEGER, INTENT(inout) :: flag    ! PDAF status flag
```

Task to be implemented:

- Fill `ens` with ensemble of initial model states

Assimilate_PDAF

Ensemble Forecast and Analysis Steps

calls `PDAF_assimilate`

- checks whether ensemble integration reached time for analysis step

- **If false:**
  - return to model and continue integration
- **If true:**
  - Write forecast fields into state vectors (call-back routine)
  - Compute analysis step of chosen filter
  - Set length of next forecast phase (call-back routine)
  - Write state vectors into model field arrays (call-back routine)

Finalize_PDAF

Clean-up of Data Assimilation Program

Clean-up at end of program

- Display timing and memory information for PDAF
- Deallocate arrays inside PDAF

Calls to

`PDAF_print_info` (memory and timing info)

`PDAF_deallocate` (deallocate arrays)

Filter analysis implementation

Operate on state vectors

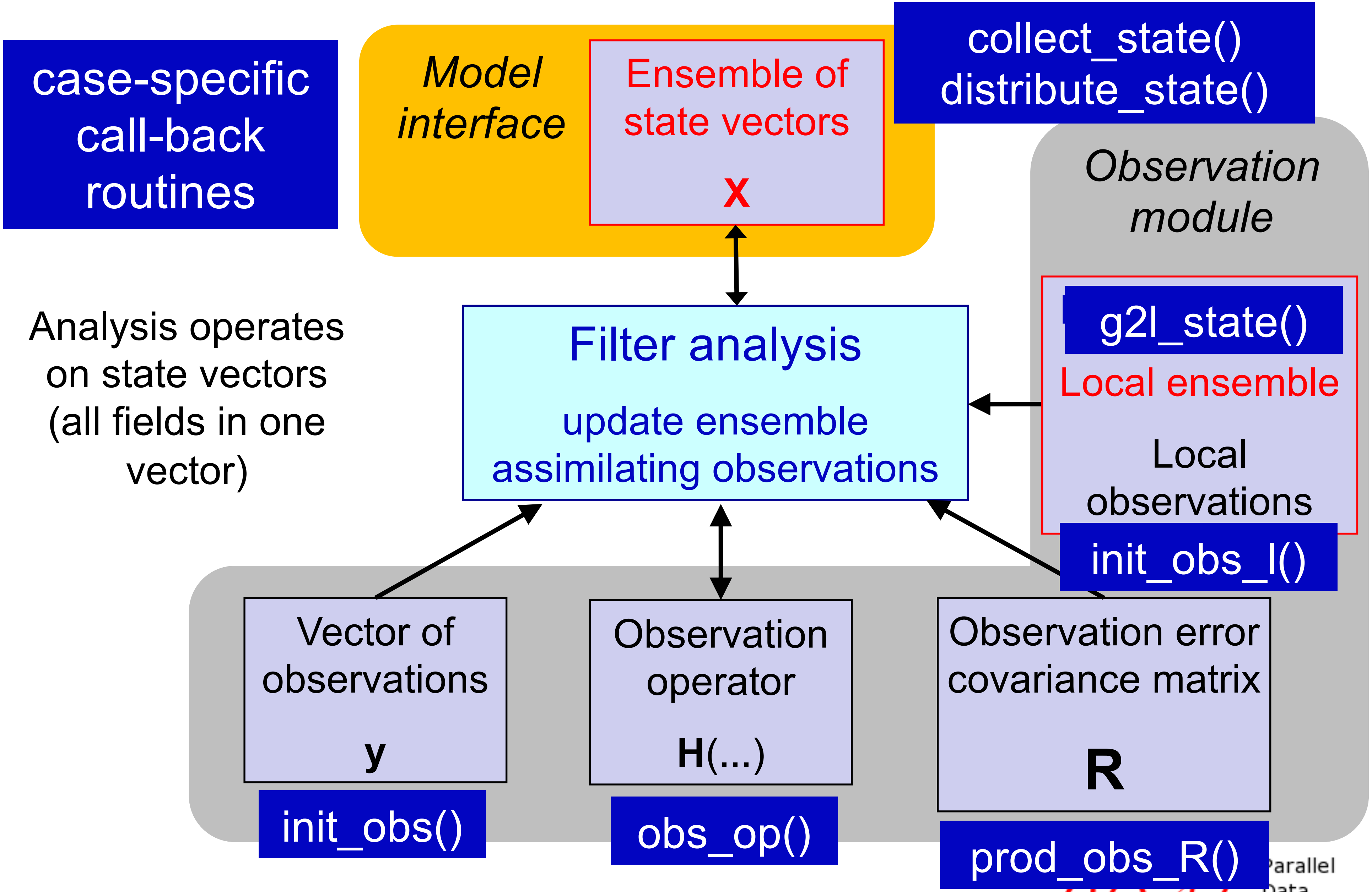
➢ Write all model fields into a 1-dimensional vector

- Filter doesn't know about 'fields'
- Computationally most efficient
- Call-back routines for
  - Transfer between model fields and state vector
  - Observation-related operations
  - Localization operations

For forecast

- Transfer data from state vector to model fields

Ensemble Filter Analysis Step



PDAF Parallel Data Assimilation Framework

Current algorithms in PDAF

PDAF originated from comparison studies of different filters

Filters and smoothers

- EnKF (Evensen, 1994 + perturbed obs.)
- ETKF (Bishop et al., 2001)
- SEIK filter (Pham et al., 1998)
- ESTKF (Nerger et al., 2012)
- NETF (Toedter & Ahrens, 2015)

Not yet released:

- serial EnSRF
- particle filter
- EWPF

All methods include

- global and localized versions
- smoothers

Model bindings

- MITgcm, Lorenz96

Not yet released:

- NEMO

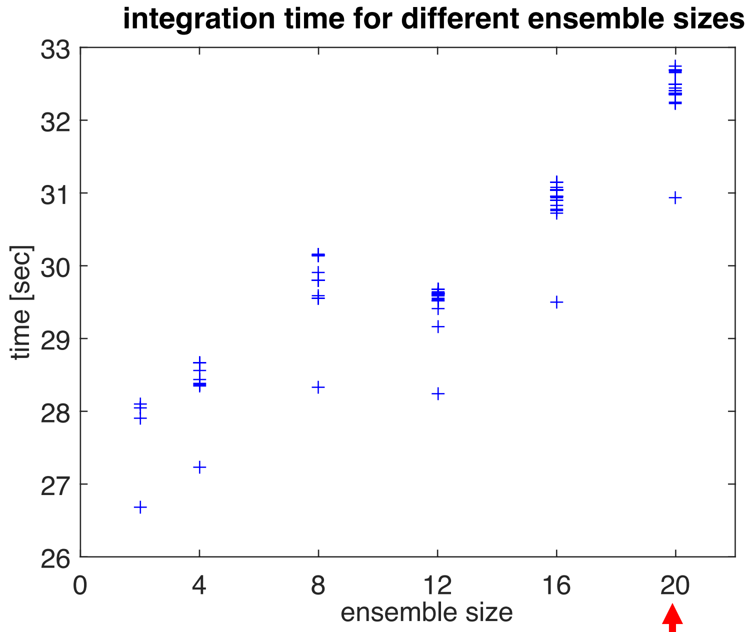
Execution times (weakly-coupled, DA only into ocean)

MPI-tasks

- ECHAM: 144
- FESOM: 384

Timings (1 day):

- Ens. forecast: 27 – 23 sec
- Analysis step: 0.5 – 0.9 sec



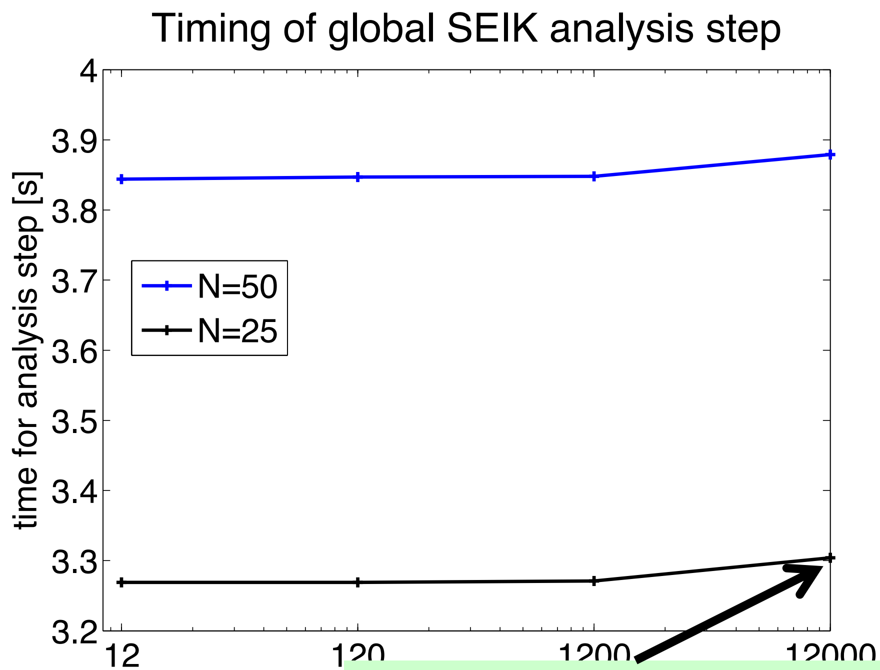
10,560 processor cores

A remaining issue:

- Increasing integration time with growing ensemble size (only 16% due to more parallel communication)
- some variability in integration time over ensemble tasks
- Need optimal distribution of programs over compute nodes/racks (here set up as ocean/atmosphere pairs)

Very big test case

- Simulate a “model”
- Choose an ensemble
  - state vector per processor: 10^7
  - observations per processor: $2 \cdot 10^5$
  - Ensemble size: 25
  - 2GB memory per processor
- Apply analysis step for different processor numbers
  - 12 – 120 – 1200 – 12000



State dimension: 1.2e11

Observation dimension: 2.4e9

- Very small increase in analysis time (~1%) (Ideal would be constant time)
- Didn’t try to run a real ensemble of largest state size (no model yet)

Implementation concept of PDAF

For ensemble data assimilation with PDAF

- Augment program for ensemble data assimilation
- Assimilation methods provided by PDAF
- Model-binding routines required
  - provided for Lorenz96 and for MITgcm for test case
  - easy to code yourself

Next look into an example

pdaf@awi.de

Slides are available online:

http://pdaf.awi.de

PDAF Parallel Data Assimilation Framework

3

Hands-on Example:
Build an Assimilation System with PDAF

Get the tutorial code

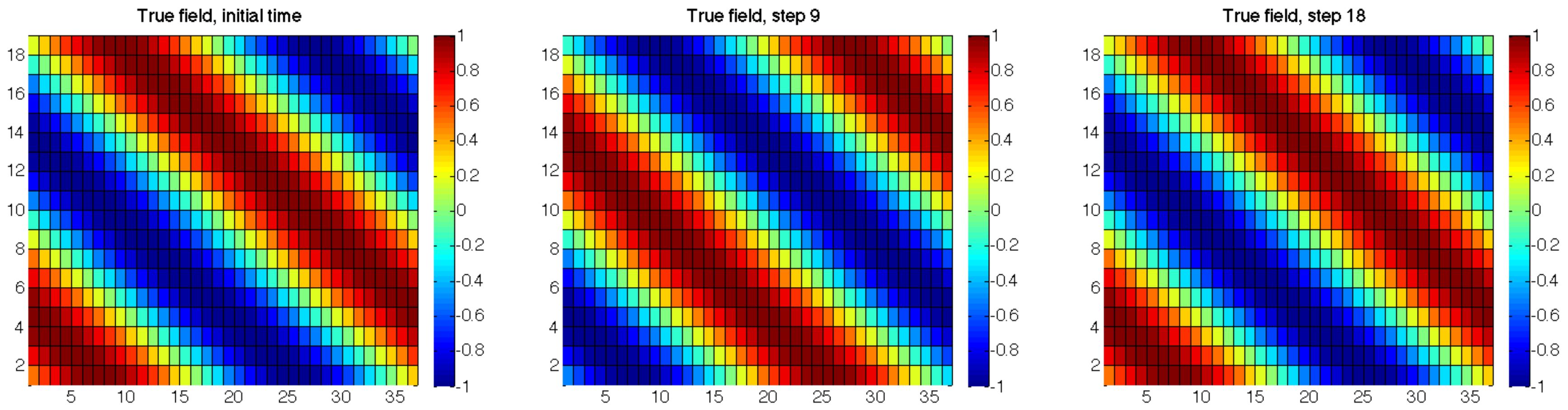
Download the tutorial

Directory layout:

`make.arch`	- build configurations
`src`	- source files
`tutorial`	
`    online_2D_serial`	
`        model`	- serial model code
`        model_coupled_to_pdaf`	- final assimilation code
`    pdaf`	- code to be added to the model
`    online_2D_serial.noMPI`	- alternative code without MPI

2D „Model“

- Simple 2-dimensional grid 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`



Parallel Data Assimilation Framework

General program structure: `model/main.f90`

```
program main
    initialize
    integrate
end program
```

`initialize` initialize model information:

- set dimensions
- allocate model field array
- read initial field

`integrate` perform time stepping

- shift model field
- write new model field

No parallelization!

Files in the tutorial directories

The model source code consists of the following files (`model/`):

- `main.F90`
- `mod_model.F90`
- `initialize.F90`
- `integrate.F90`
- `Makefile`

Files in the tutorial directories

The PDAF coupling code consists of (`pdaf/`)

- interface subroutines (called from the model code)
  - `init_parallel_pdaf.F90`
  - `init_pdaf.F90`
  - `assimilate_pdaf.F90`
  - `finalize_pdaf.F90`
- user subroutines (called from the PDAF library), eg.
  - `collect_state_pdaf.F90`
- "supporting" modules and subroutines (used in the interface and user subroutines), eg.
  - `mod_assimilation.F90`
  - `init_pdaf_parse.F90`

Running the tutorial model

- cd to `tutorial/online_2D_serialmodel/model`
- Set environment variable `PDAF_ARCH`

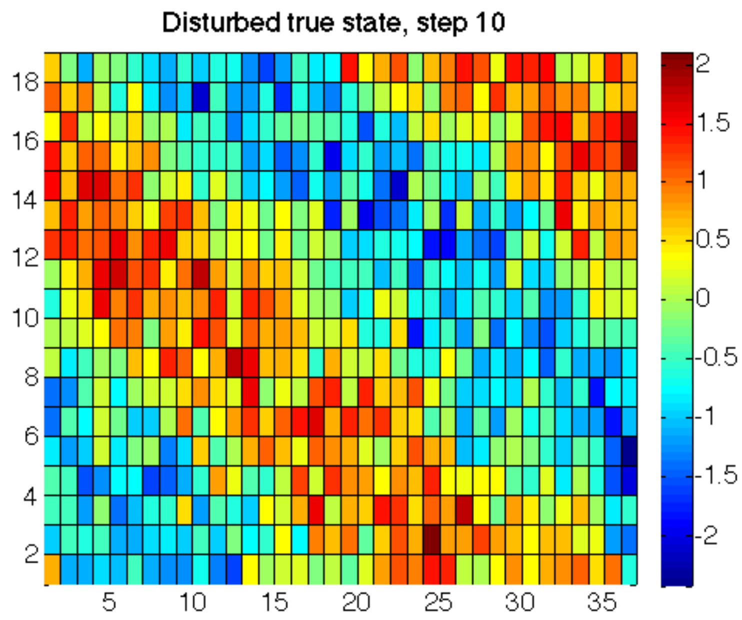
  `export PDAF_ARCH=linux_gfortran_openmpi`
- Run `make`
- Run the model with `./model`

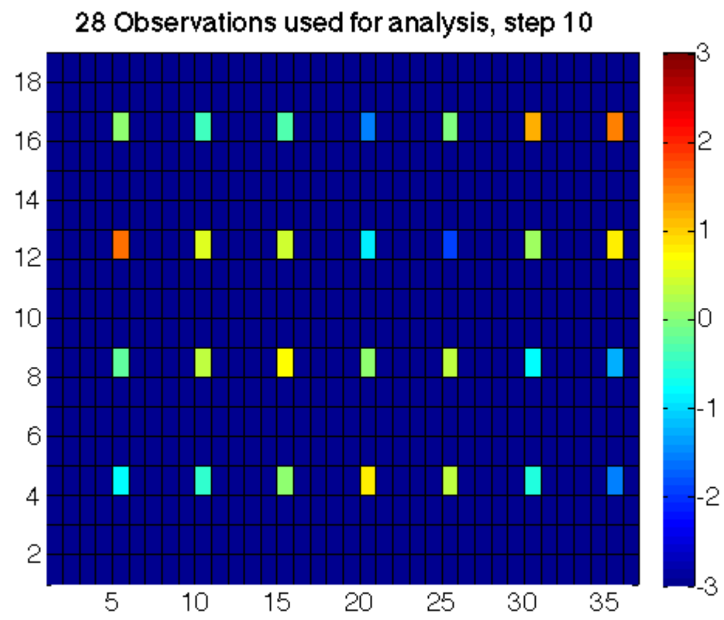
- Inputs are read in from `tutorial/inputs_online`
- Outputs are written in `tutorial/online_2D_serialmodel/model`

  eg. `true_step10.txt`

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





Coupling the model to PDAF: Online mode

- Combine model with PDAF into single program
  - modify `Makefile` to build `model_pdaf`

- Add 4 subroutine calls:

  `init_parallel_pdaf` - add parallelization
  `init_pdaf` - initialize assimilation
  `assimilate_pdaf` - perform assimilation
  `finalize_pdaf` - clean up

- Implement user subroutines, e.g. for
  - observation operator
  - initialization of observation vector
  - transfer between state vector and model fields

http://pdaf.awi.de/trac/wiki/OverviewOfUserRoutinesWithDefaultNames

Online coupling: Parallelization

- Online coupling avoids writing to disk to exchange state vectors between the model and PDAF
- Add MPI to the model to run several model instances in parallel
- Run the parallel version with

```
mpirun -np <n> ./model_pdaf ...
```

- *Alternative*: PDAF's "flexible" approach:
  http://pdaf.awi.de/ModifyModelForEnsembleIntegration
  - cd to `tutorial/online_2D_serialmodel.noMPI/model`

Files to copy from `pdaf` to `model`

`init_parallel_pdaf.F90`
- `mod_parallel_pdaf.F90`
- `parser_mpi.F90`

parallelization

PDAF interface subroutine - called from the model

helper module/subroutine for the interface

PDAF user subroutine - called from PDAF library

`finalize_pdaf.F90` — clean up

`init_pdaf.F90`
- `mod_assimilation.F90`
- `init_pdaf_info.F90`
- `init_pdaf_parse.F90`
- `init_ens.F90`

initialization

- `next_observation_pdaf.F90`
- `distribute_state_pdaf.F90`

ensemble forecast

- `prepoststep_ens_pdaf.F90`

post step

... (continued on next slide)

Files to copy from `pdaf` to `model`

... (continued from previous slide)

```
assimilate_pdaf.F90
  collect_state_pdaf.F90
  init_dim_obs_pdaf.F90
  obs_op_pdaf.F90
  init_obs_pdaf.F90
  prodrinva_pdaf.F90
```

analysis step

- Each file contains a short summary what the subroutine does

Files to be adapted in `model`

main.F90	-	add calls to PDAF interface
integrate.F90	-	add calls to PDAF interface
Makefile	-	add linking to PDAF library, PDAF interface and user subroutines

- *Reference solutions for the modified files are in* *model_coupled_to_pdaf*
- When complete, run make again
- Then run

```
mpirun -np 9 ./model_pdaf -dim_ens 9
```

- Outputs are written to

ens_<i>_step<j>_for.txt

ens_<i>_step<j>_ana.txt

This runs a filter without localization with ensemble size 9

Plotting

- When your coupling is working, lookt at the results
- With Matlab/Octave you can use

```
load ens_01_step02_for.txt
pcolor(ens_01_step02_for)
```

- Or use the Python scripts

```
./plot_file.py ens_<i>_step<j>_for.txt
./plot_ens.py <i> <j>
```

More PDAF experiments

- Find PDAF command line parameters in

```
./pdaf/init_pdaf_parse.F90
```

- Try for example

```
mpirun -np 4 ./model_pdaf -dim_ens 4
```

(this runs a filter (ESTKF) without localization with ensemble size 4; it gives a worse result than ensemble size 9)

```
mpirun -np 9 ./model_pdaf -dim_ens 9 -filtertype 7
```

(this runs a filter (LESTKF) with localization and localization radius 0, i.e. correcting only at observed grid points)

```
mpirun -np 9 ./model_pdaf -dim_ens 9 -filtertype 7
  -local_range 5
```

(this runs a filter (LESTKF) with localization and localization radius of 5 grid points)

Feedback, Questions, more code, …

Full PDAF package contains

- more tutorial code, more filters, and the fully implemented Lorenz-96 model and MITgcm model binding

Web site provides an extensive tutorial for self-study

For further questions

- Contact us at pdaf@awi.de
- Poster A.14, Friday 14:00–15:45 (L. Nerger)

pdaf@awi.de

Slides are available online:

http://pdaf.awi.de

PDAF Parallel Data Assimilation Framework