PDAF Parallel Data Assimilation Framework

Short Course SC5.14

Practical Ensemble Data Assimilation with the Parallel Data Assimilation Framework

Lars Nerger[1], Wolfgang Kurtz[2],
Nabir Mamnun[1], Qi Tang[1,3], Gernot Geppert[4]

1: Alfred Wegener Institute, Bremerhaven, Germany

2: Leibnitz Supercomputing Center, Garching, Germany

3: Institute of Geographic Sciences and Natural Resources Research, CAS, Beijing, China

4: Deutscher Wetterdienst (DWD), Offenbach, Germany

http://pdaf.awi.de

Hands-on code: **http:/pdaf.awi.de/EGU2021**

The Short Course – Overview

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

For hands-on example:
(try assimilation yourself)

code available at **http://pdaf.awi.de/EGU2021**

PDAF Parallel Data Assimilation Framework

For hands-on example:

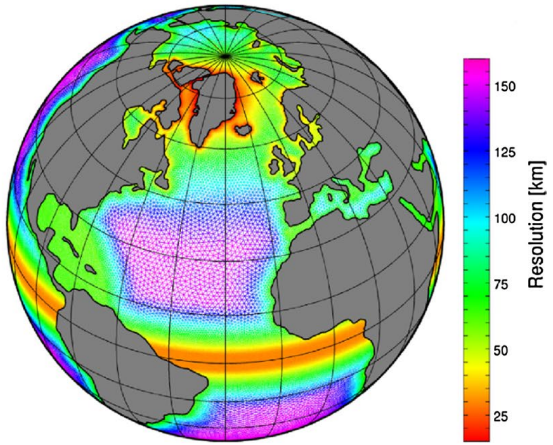
code available at **http://pdaf.awi.de/EGU2021**

1

Ensemble Data Assimilation

Data Assimilation

Models



Observations

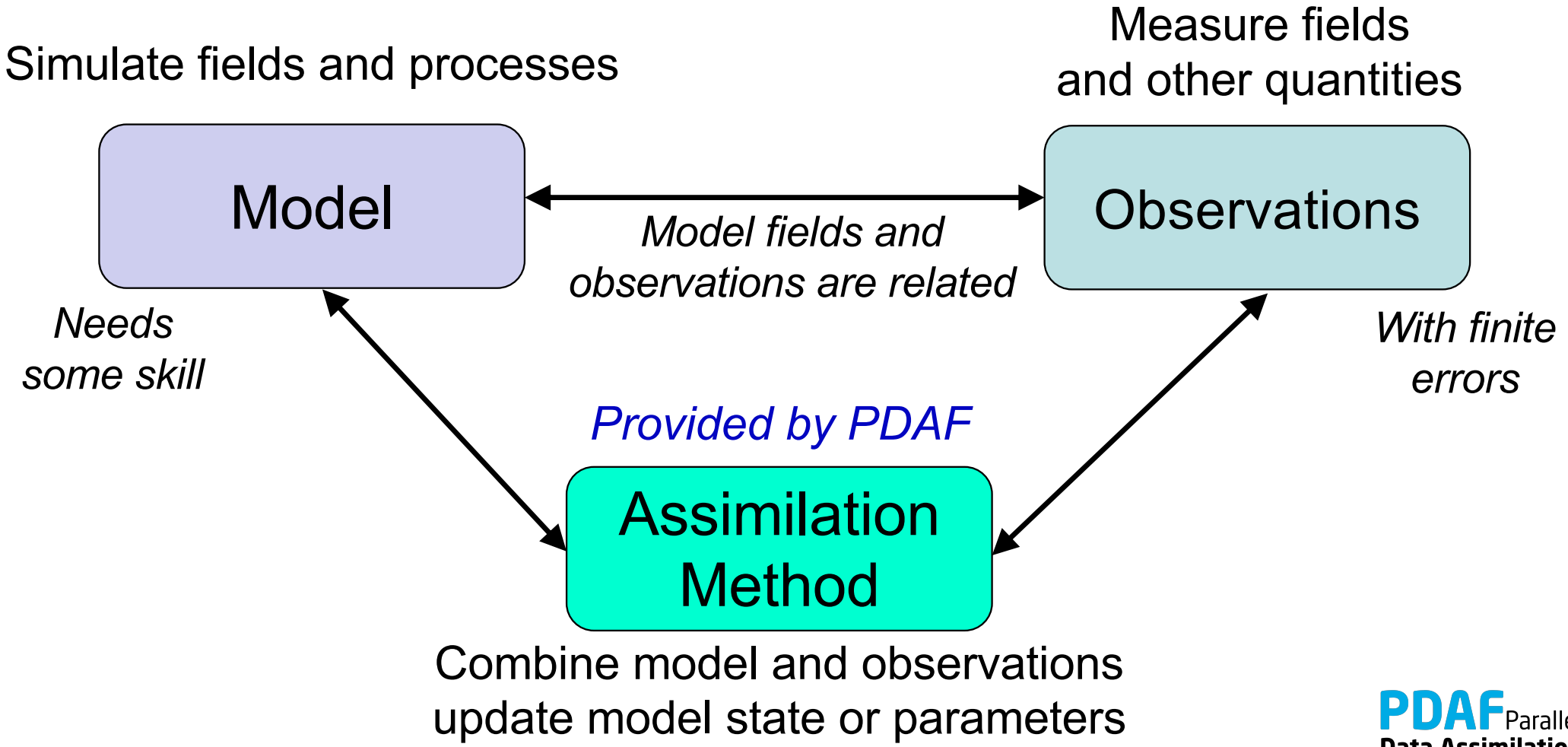
Data Assimilation

Combine both sources of information quantitatively

- → Go beyond just comparing model and observations
- → Provide direct link between observations and models
- → Allow models to learn from data
- → Enhance data products

PDAF Parallel Data Assimilation Framework

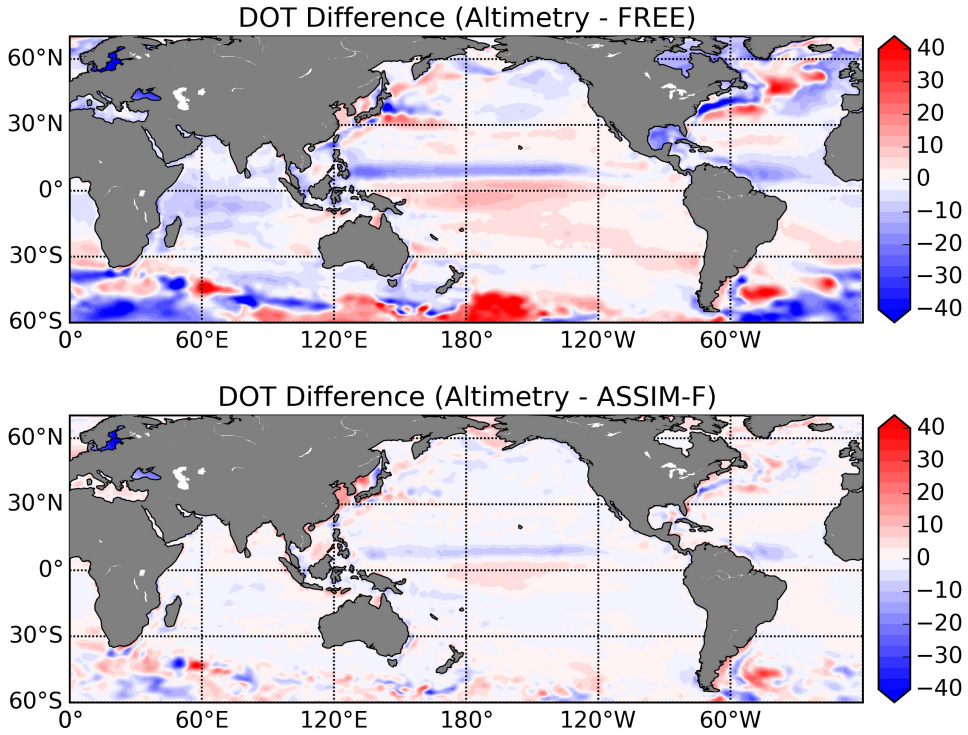
Components of an Assimilation System



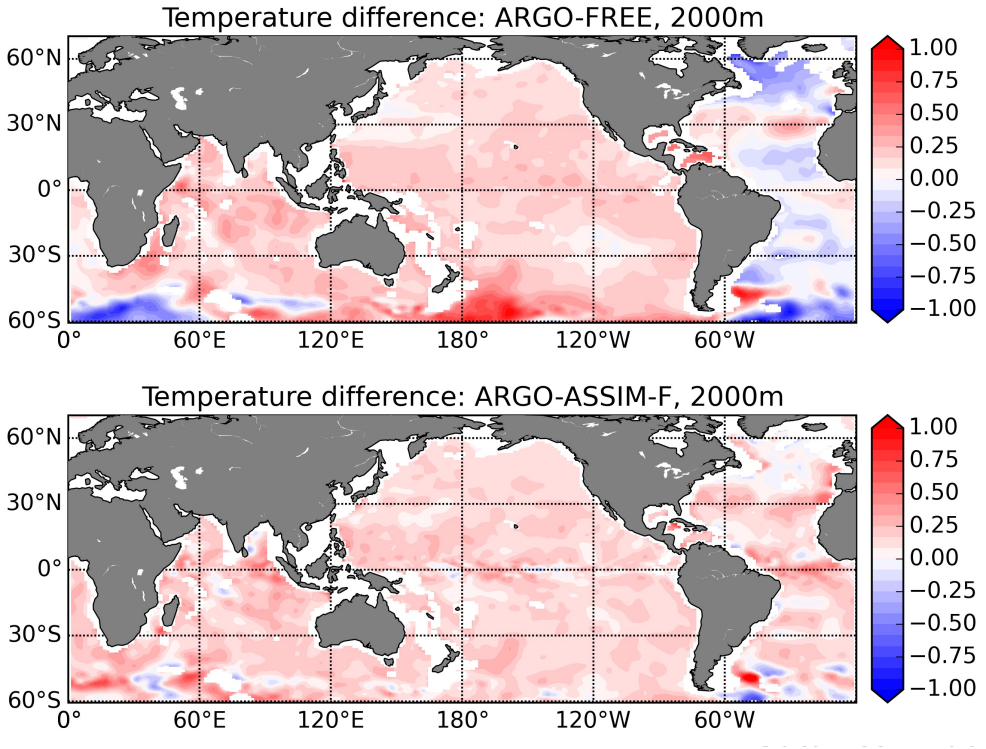
Example: Assimilation of sea surface height data

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

lel
on
Framework

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, ice strength, …)
  - 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

Ensemble Data Assimilation

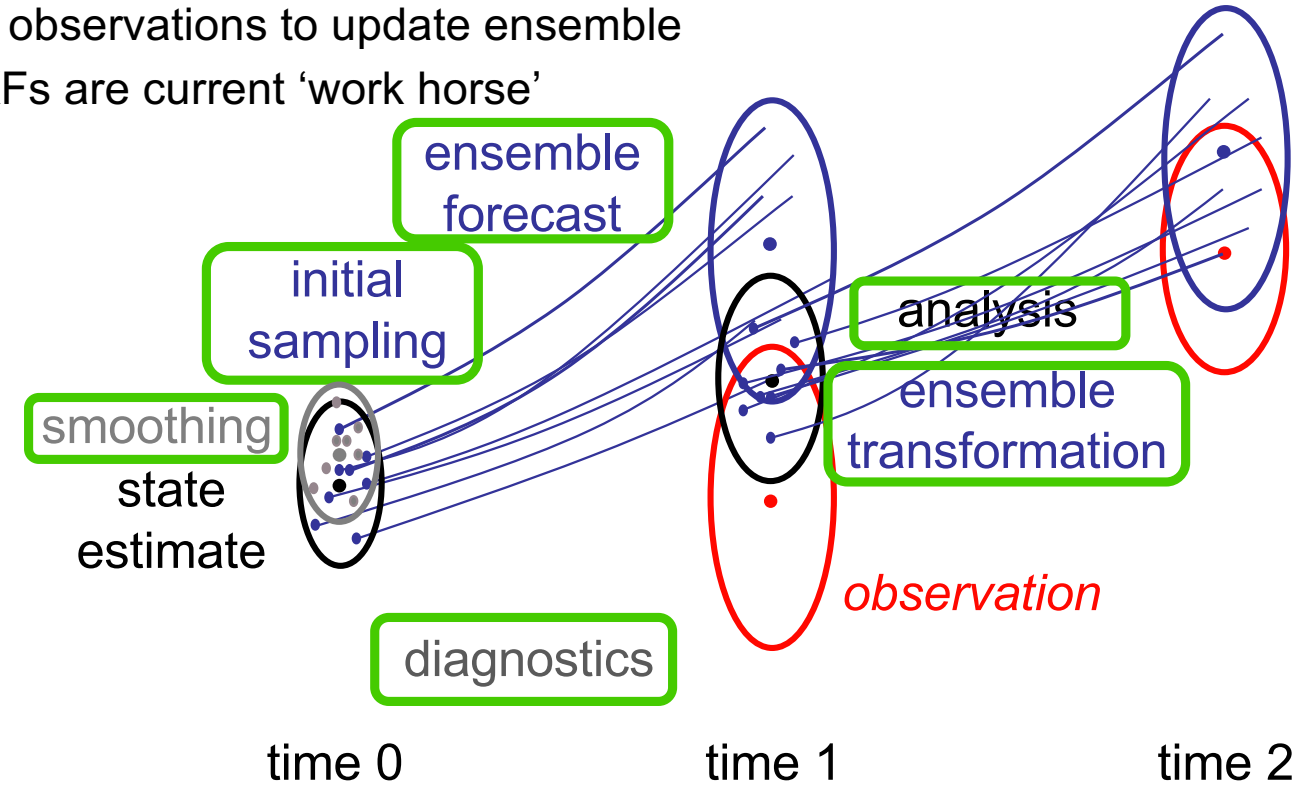
Ensemble Kalman Filters & Particle Filters

→ Use ensembles to represent state and uncertainty

→ Propagate ensemble using numerical model

→ Use observations to update ensemble

→ EnKFs are current 'work horse'



These steps can be implemented in a generic way

We can provide software including the algorithms

→ PDAF

PDAF Parallel Data Assimilation Framework

For hands-on example:

code available at **http://pdaf.awi.de/EGU2021**

2

Implementation Concept of PDAF

(Parallel Data Assimilation Framework)

PDAF Parallel Data Assimilation Framework

PDAF – Community Ensemble Data Assimilation Software

PDAF - Parallel Data Assimilation Framework

- a program library for ensemble data assimilation
- provides support for parallel ensemble forecasts
- provides filters and smoothers - fully-implemented & parallelized
  (EnKF, LETKF, LESTKF, NETF, PF … easy to add more)
- easily usable with (probably) any numerical model
  (coupled to e.g. NEMO, MITgcm, FESOM, HBM, MPI-ESM, SCHISM/ESMF)
- run from laptops to supercomputers (Fortran, MPI & OpenMP)
- Usable for real assimilation applications and to study assimilation methods
- ~500 registered users; community contributions

Open source:
Code, documentation, and tutorial available at

http://pdaf.awi.de

PDAF Parallel Data Assimilation Framework

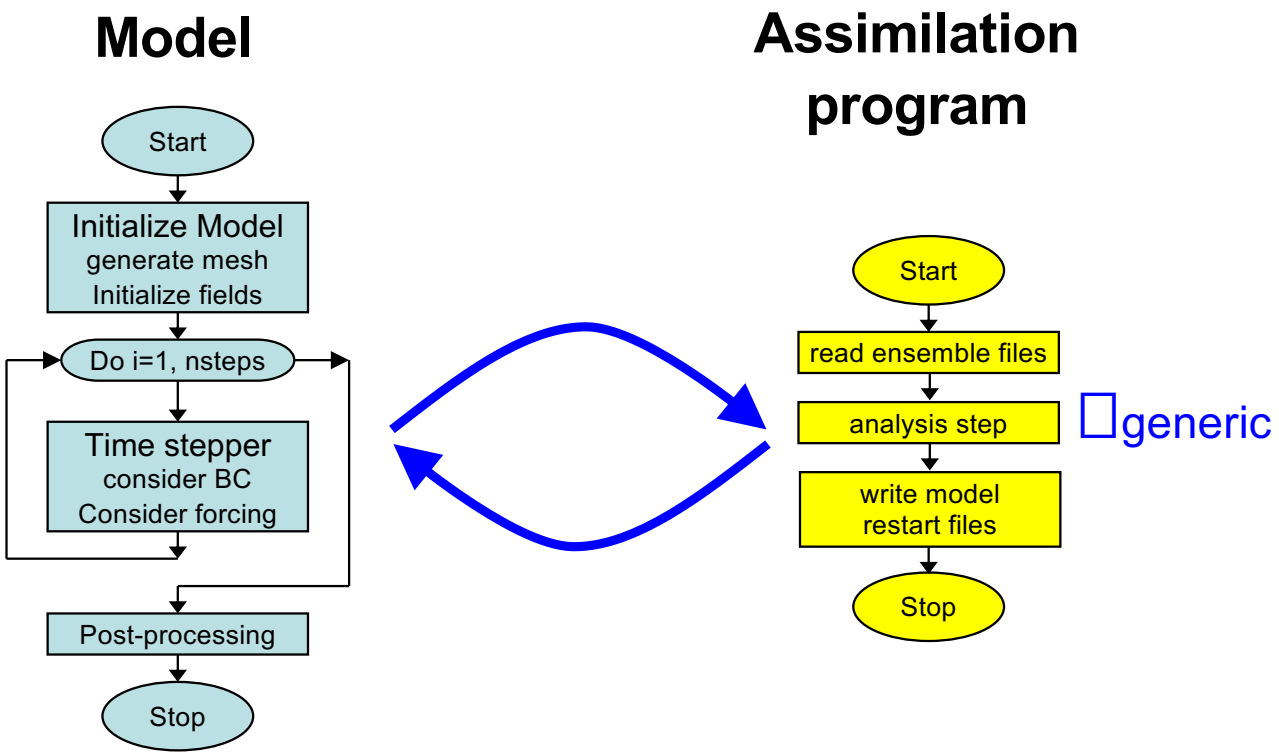
L. Nerger, W. Hiller, Computers & Geosciences 55 (2013) 110-118

Offline coupling – separate programs

PDAF supports offline coupling

-

we don't cover this here



For each ensemble state

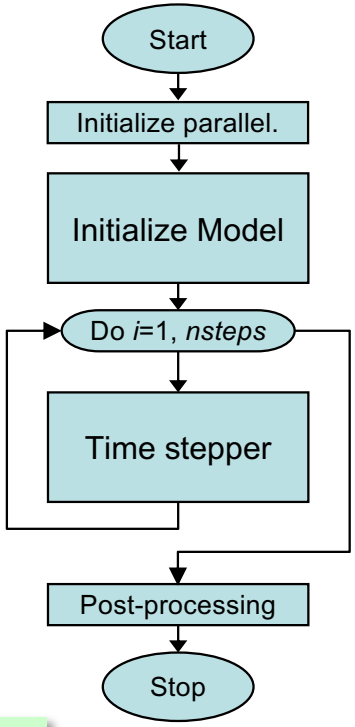
- Initialize from restart files
- Integrate
- Write restart files

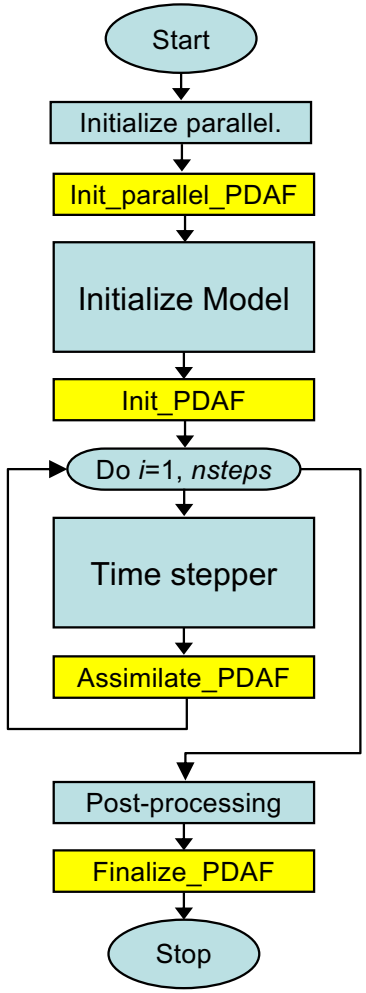
- Read restart files (ensemble)
- Compute analysis step
- Write new restart files

PDAF Parallel Data Assimilation Framework

Online coupling - Augmenting a Model for Data Assimilation

Model





Extension for data assimilation

revised parallelization enables ensemble forecast

Data assimilation: run model with additional options

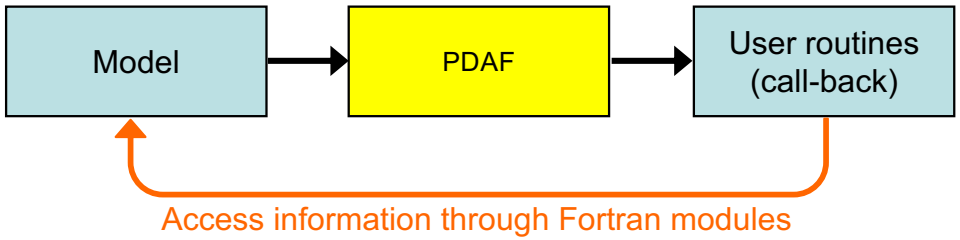
plus:
Possible model-specific adaption

e.g. in NEMO or ECHAM: treat leap-frog time stepping

PDAF Parallel Data Assimilation Framework

PDAF interface structure

- Interface routines call PDAF-core routines
- PDAF-core routines call case-specific routines provided by user (included in model binding set)
  - Provide information to PDAF at time when needed
- 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



Ensemble-Parallelization of Assimilation Program

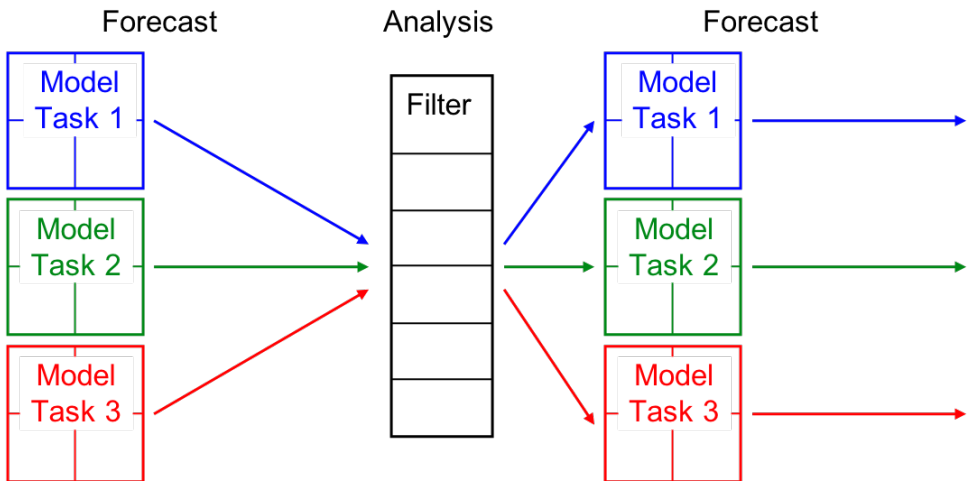
Modify model to become an 'ensemble model'

We use MPI (Message Passing Interface)

- standard used by most large-scale models

Only need to prepare `init_parallel_pdaf` once for a model

- There is a template that works with most models



Init_PDAF

Initialize the Assimilation

Set parameters, for example

- select filter
- set ensemble size

Calls `PDAF_init`

- initialization routine of framework
- provide parameters according to interface
- PDAF_init calls the user-provided ensemble initialization routine

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:**
  - Compute analysis step of chosen filter

Finalize_PDAF

Clean-up of Data Assimilation Program

Clean-up at end of program (optional)

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

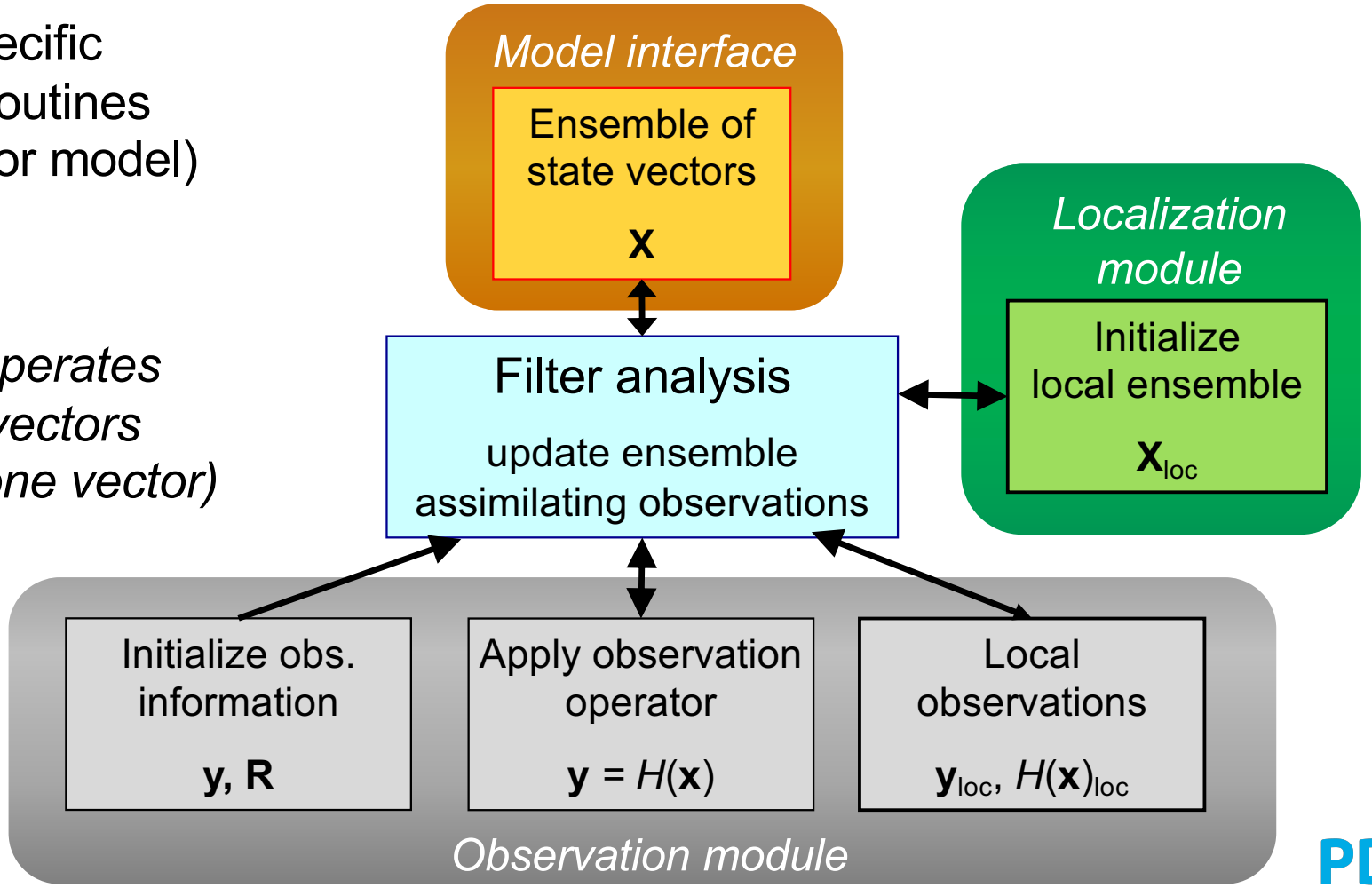
Calls to

`PDAF_print_info`	(memory and timing info)
`PDAF_deallocate`	(deallocate arrays)

Implementing the Ensemble Filter Analysis Step

case-specific
call-back routines
(implement for model)

Analysis operates
on state vectors
(all fields in one vector)



Implementation concept of PDAF

For ensemble data assimilation with PDAF

- Augment program for ensemble data assimilation (or use separate assimilation program)
- Assimilation methods provided by PDAF
- Model-binding routines required
  - provided MITgcm for test case, and AWI-CM/FESOM
  - easy to code yourself
  - Full implementation for Lorenz96, Lorenz65 and 2d toy model

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:
An Assimilation System built with PDAF

Tutorial code available at
http://pdaf.awi.de/EGU2021

Get the tutorial code

Download the tutorial from **http://pdaf.awi.de/EGU2021**

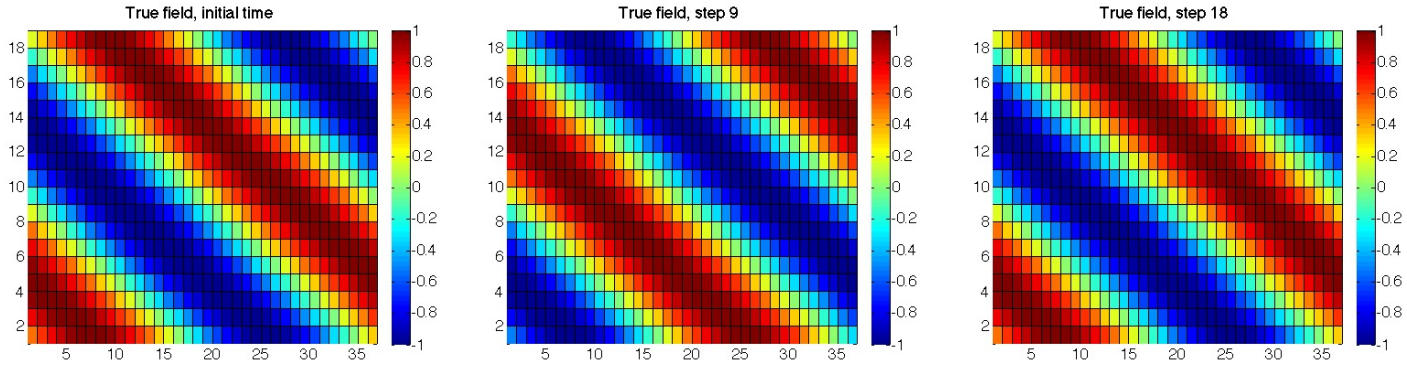
Directory layout:

```
make.arch/                          - build configurations
src/                                - source files
tutorial/
   online_2D_serial/
      model/                        - serial model code
      model_coupled_to_pdaf/        - final assimilation code

   online_2D_serial.noMPI/          - alternative code without MPI
```

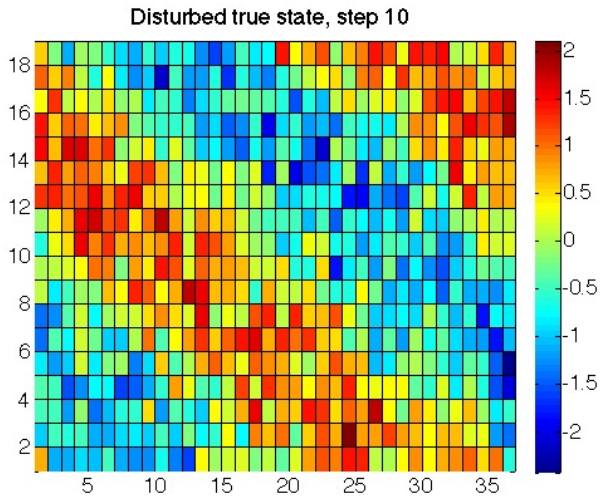
2D „Model“

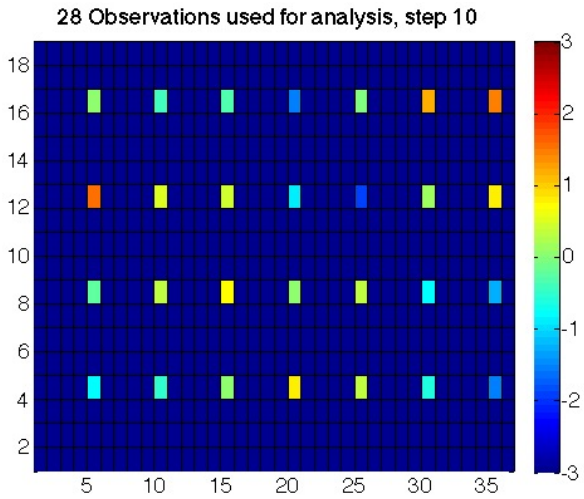
- Simple 2-dimensional grid domain
- 36 x 18 grid points (longitude x latitude)
- True state: sine wave in diagonal direction
  (periodic for consistent time stepping)
- Simple time stepping:
  Shift field in vertical direction one grid point per time step
- Output to text files (18 rows) – `true_step*.txt`



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





PDAF Parallel Data Assimilation Framework

General program structure: `model/main.f90`

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

initialize model information:
- set dimensions
- allocate model field array
- read initial field

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 (`model_coupled_to_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 tutorial/online_2D_serialmodel/model`
- Run `make PDAF_ARCH=linux_gfortran`
  - for gcc/gfortran 10 use `linux_gfortran10`
- 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`
- Plot with python eg.: `../../plotting/plot_file.py true_step18.txt`

- Note: One can also specify PDAF_ARCH as environment variable, e.g.

  `export PDAF_ARCH=linux_gfortran`

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 eg. with

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

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

Adding files for the assimilation – ensemble runs

```
init_parallel_pdaf.F90
  mod_parallel_pdaf.F90

finalize_pdaf.F90

init_pdaf.F90
  mod_assimilation.F90
  parser_mpi.F90
  init_pdaf_parse.F90
  init_ens_pdaf.F90
  next_observation_pdaf.F90
  distribute_state_pdaf.F90
  prepoststep_ens_pdaf.F90
```

parallelization

clean up

initialization

ensemble forecast

post step

PDAF interface subroutine - called from the model

helper module/subroutine for the interface

PDAF user subroutine - called from PDAF library

Routines are generally short – PDAF provides templates to simplify implementation

... (continued on next slide)

Directory: online_2D_serialmodel/model_coupled_to_pdaf/

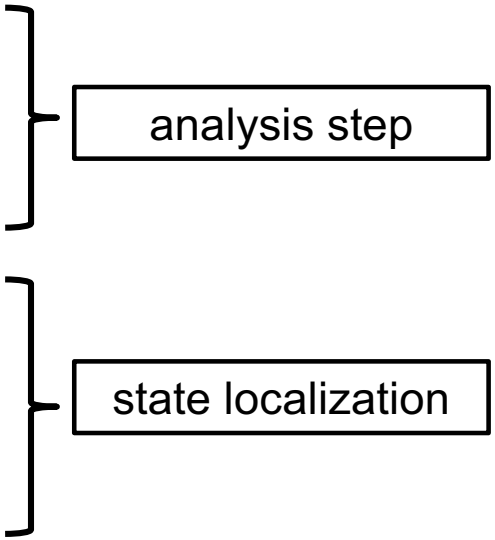
PDAF Parallel Data Assimilation Framework

Adding files for the assimilation – analysis step

... (continued from previous slide)

```
assimilate_pdaf.F90
  collect_state_pdaf.F90
  callback_obs_pdafomi.F90
  obs_A_pdafomi.F90

  init_n_domains_pdaf.F90
  init_dim_l_pdaf.F90
  g2l_state_pdaf.F90
  l2g_state_pdaf.F90
```



PDAF interface subroutine - called from the model

PDAF user subroutine - called from PDAF library

PDAF-OMI observation handling – called from PDAF library

Routines are generally short – PDAF provides templates to simplify implementation

- Each file contains a short summary what the subroutine does
- Templates also describe the required operations

Modified files in `model_coupled_to_pdaf`

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

- Run `make PDAF_ARCH=linux_gfortran_openmpi`
- Then run

```
mpirun -np 5 ./model_pdaf -dim_ens 5
```

- 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 5

Note: PDAF needs the numerical libraries LAPACK and BLAS (available as standard Linux packages)

Observation handling – PDAF-OMI

`obs_A_pdafomi.F90` - Observation module containing 3 subroutines:

- `init_dim_obs_A` - initialize observation information
- `obs_op_A` - apply observation operator
- `init_dim_obs_l_A` - find local observations

An observation module contains one observation type

`callback_obs_pdafomi.F90` - Interface routine called by PDAF

- contains the calls to the observation-specific routines
- contains deallocation routine `deallocate_obs_pdafomi`

Other observation types

`obs_B_pdafomi.F90`
- Activate with '-assim_B .true.'

`obs_C_pdafomi.F90`
- Activate with '-assim_C .true.'

(obs_C uses observation operator with interpolation)

PDAF Parallel Data Assimilation Framework

Plotting

- Plotting the results
- With Matlab/Octave you can use

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

- Or use the Python scripts

```
../../plottig/plot_file.py ens_<i>_step<j>_ana.txt
../../plotting/plot_ens.py <i> <j>
```

(Python plotting requires numpy, matplotlib, and argparse)

More PDAF experiments

- Find PDAF command line parameters in

  `./model_coupled_to_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)

Current algorithms in PDAF

PDAF originated from comparison studies of different filters

Filters and smoothers - *global and localized versions*

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

Not yet released:
- Ensemble 3D-Var
- serial EnSRF
- EWPF

Model bindings

- MITgcm
- AWI-CM / FESOM

Toy models

- Lorenz-96 / Lorenz-63

Feedback, Questions, more code, …

Full PDAF package contains

- more tutorial code, more filters
- the fully implemented Lorenz-96/63 models
- MITgcm and FESOM/AWI-CM model bindings

Web site provides an extensive tutorial for self-study

For further questions

- Contact us at pdaf@awi.de

pdaf@awi.de

Slides are available online:
http://pdaf.awi.de

PDAF Parallel Data Assimilation Framework