PDAF Tutorial

The tutorial model, observations and initial ensemble

http://pdaf.awi.de

PDAF Parallel Data Assimilation Framework

V1.0 – 2020-11-25

Implementation Tutorial for PDAF online with serial model

We demonstrate the implementation

of an online analysis step with PDAF

with a model that is parallelized

using the template routines provided by PDAF

The example code is part of the PDAF source code package
downloadable at http://pdaf.awi.de

(This tutorial is compatible with PDAF V1.16 and later)

PDAF Parallel Data Assimilation Framework

Implementation Tutorial for PDAF online / parallel model

This is just an example!

For the complete documentation of PDAF's interface

see the documentation

at http://pdaf.awi.de

Tutorial implementations

Files are in the PDAF package

Directories:

Model without parallelization

`/tutorial/online_2D_serialmodel`

Model without parallelization (domain-decomposition)

`/tutorial/online_2D_parallelmodel`

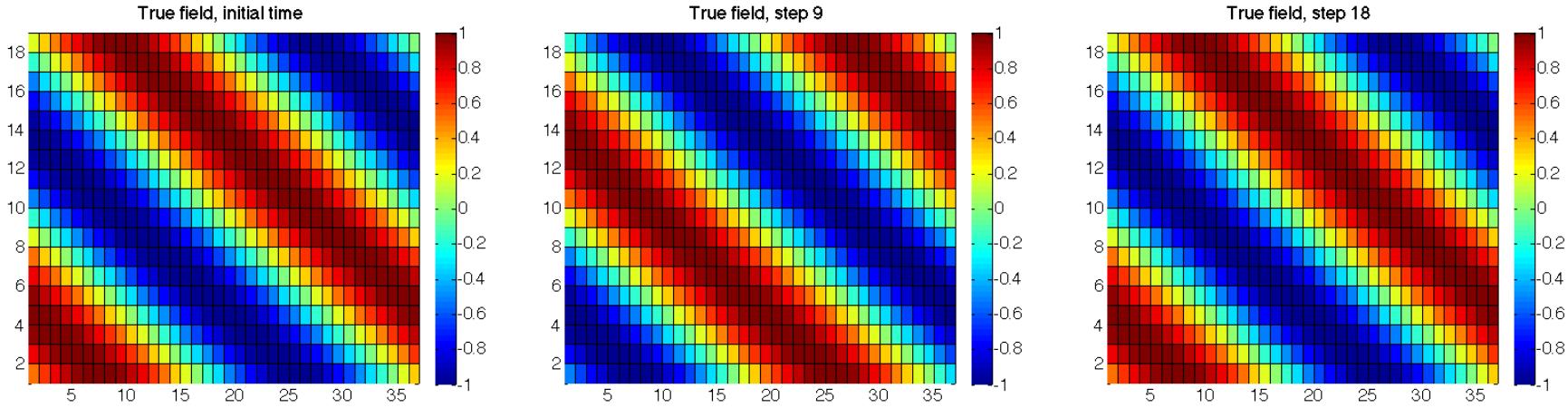
- Fully working implementations of user codes
- Only need to specify the compile settings (compiler, etc.) by environment variable PDAF_ARCH. Then compile with '`make`'.

0a) The model without parallelization

`/tutorial/online_2D_serialmodel`

2D „Model“

- A single field in 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
- Stored in text files (18 rows) – `true_step*.txt`



`Model`: General program structure

```
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!

`Model`: Shared variables

Shared variables are declared in Fortran module (mod_model.F90)

```
MODULE mod_model

…

  INTEGER :: nx, ny              ! Size of 2D grid
  INTEGER :: total_steps         ! Total number of time steps
  REAL, ALLOCATABLE :: field(:,:)          ! Model field

END MODULE mod_model
```

- Included with 'use' in `initialize` and `integrate`

`Model`: Files

The model source code consists of the following files:

- mod_model.F90
- main.F90
- initialize.F90
- integrate.F90

Running the tutorial model

- cd to `/tutorial/online_2D_serialmodel`
- Set environment variable `PDAF_ARCH` or set it in Makefile (e.g. `linux_gfortran`)
- Compile by running `'make model'`
- Run the program with `./model`

- Inputs are read in from `/tutorial/inputs_online`
- Outputs are written in `/tutorial/online_2D_serialmodel`
- Plot result, e.g with 'octave':

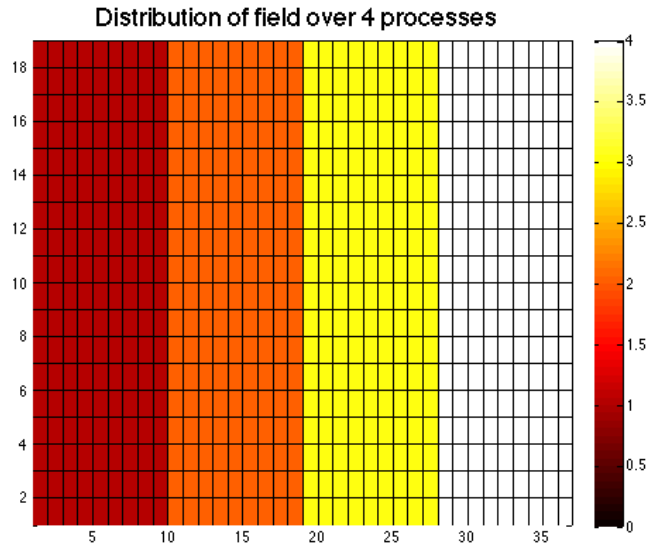
```
Load true_step10.txt

Pcolor(true_step10)
```

0b) The parallelized model

`/tutorial/online_2D_parallelmodel`

Model parallelization

- Same model as before, but now with parallelization
- Parallelization:
  - Distribute in direction of second index: `nx` ➜ `nx_p`
  - Each process holds a part of the model field (size ny * nx_p)
  - Disk files hold the global field; the information is distributed after reading and collected before a single process writes



PDAF Parallel Data Assimilation Framework

parallel `Model`: General program structure

```
program main
   init_parallel          initialize MPI parallelization
   initialize             initialize model information:
                              - set dimensions
                              - allocate model field array
                              - read initial field
   integrate              perform time stepping
                              - shift model field
                              - write new model field
   finalize_parallel      end MPI-parallel region
end program
```

`Model`: Shared variables

Shared variables are declared in Fortran module (mod_model.F90)

```
MODULE mod_model
…
  INTEGER :: nx, ny                ! Size of 2D grid
  INTEGER :: total_steps           ! Total number of time steps
  REAL, ALLOCATABLE :: field(:,:)  ! Model field
  INTEGER :: nx_p                  ! Local size in x-direction
END MODULE mod_model
```

- Included with 'use' in `initialize` and `integrate`

`Model`: Shared variables for parallelization

Shared variables for parallelization are declared in Fortran module (mod_parallel_model.F90)

```
MODULE mod_parallel_model
…
   INTEGER :: COMM_model   ! MPI communicator for model tasks
   INTEGER :: mype_model   ! Process rank in COMM_model
   INTEGER :: npes_model   ! Number of processes in COMM_model
   INTEGER :: mype_world   ! Process rank in MPI_COMM_WORLD
   INTEGER :: npes_world   ! Number of PEs in MPI_COMM_WORLD
   INTEGER :: MPIerr       ! Error flag for MPI
END MODULE mod_parallel_model
```

Parallel `model`: Files

The source code of the parallel model consists of the following files:

- mod_model.F90
- mod_parallel_model.F90
- main.F90
- initialize.F90
- integrate.F90

Note: One can nicely compare the source codes of the model without and with parallelization

Running the parallel tutorial model

- cd to `/tutorial/online_2D_parallelmodel`
- You need to compile with an MPI library!
- Set environment variable `PDAF_ARCH` or set it in Makefile (e.g. `linux_gfortran_openmpi`)
- Compile by running `'make model'`
- Run the program with `mpirun –np 4 ./model`
- Note: The model can be run with 2,3,4,6 or 9 processes (These numbers allow for a uniform distribution of nx=36)

- Inputs are read in from `/tutorial/inputs_online`
- Outputs are written in `/tutorial/online_2D_parallelmodel`

Differences model with and without parallelization

Serial model

- Global field dimensions nx, ny
- Global model field ‘`field(ny,nx)`’
- No particular condition for screen output
- Compile without MPI-Library

Parallel model

- Global dimensions nx, ny; process local dimension nx_p
- Global field distributed as sub-fields ‘`field_p(ny,nx_p)`’
- Screen and file output for process with ‘mype_world==0’
- Compile with MPI-Library

0c) Observations

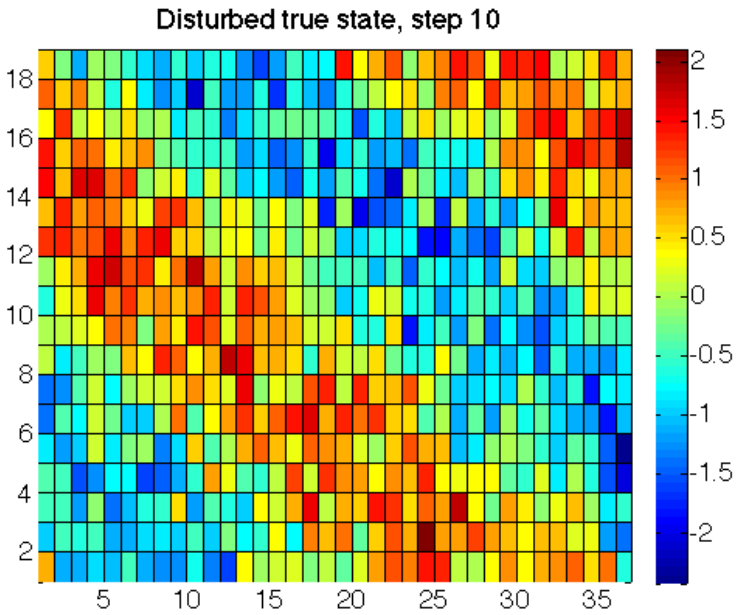
Simple assimilation problem

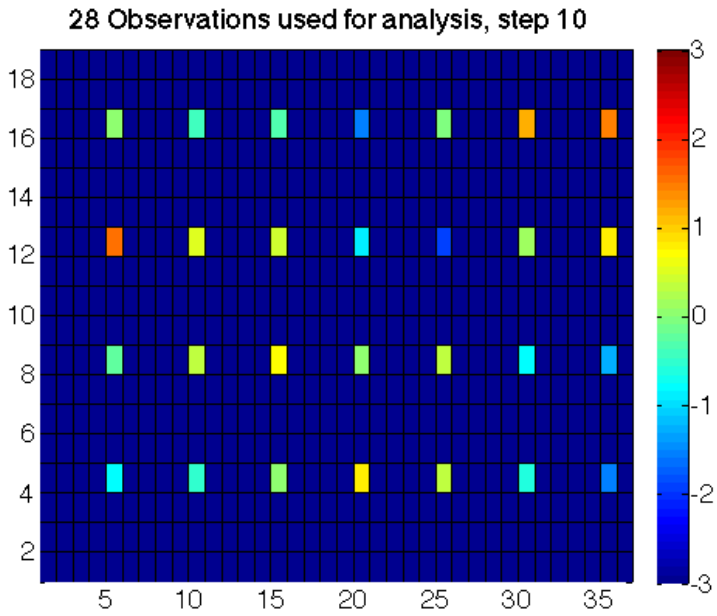
Observations for assimilation

- Direct measurements of the field
- Data gaps (i.e. data at selected grid points)
- Same error estimate for all observations
- Observation errors are not correlated
  (diagonal observation error covariance matrix)

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



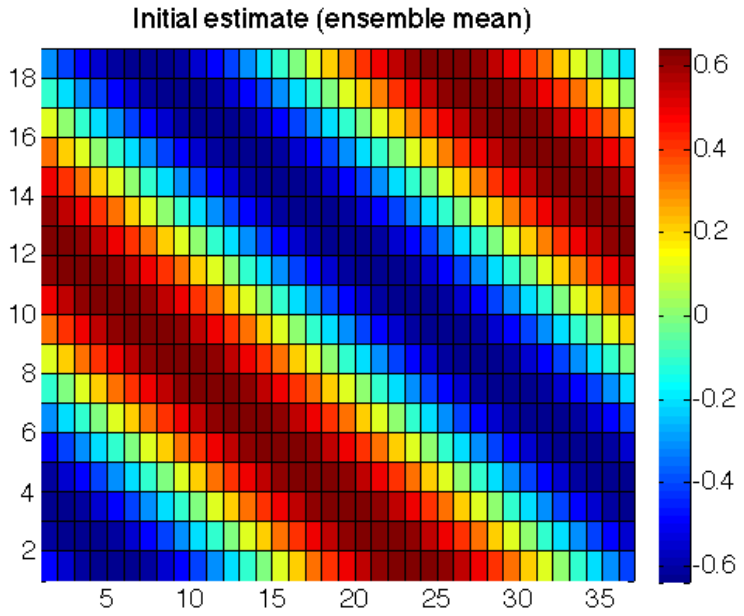


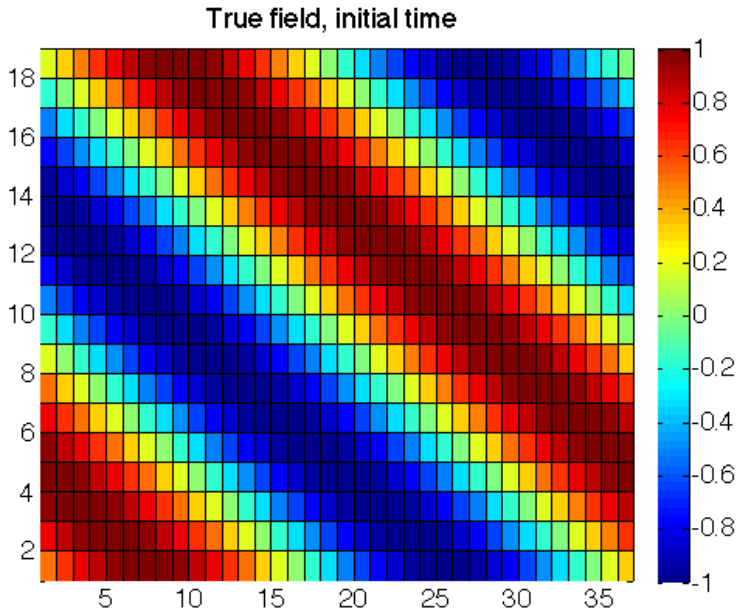
PDAF Parallel Data Assimilation Framework

0d) Initial Ensemble

Ensemble

- Prepared 9 ensemble state files
- Sine waves shifted along diagonal (truth not included)
- One text file per ensemble member – `ens_*.txt` (in `inputs_online`)





PDAF Parallel Data Assimilation Framework

Ensemble states at initial time