Tutorial T002

Ensemble Data Assimilation with the Parallel Data Assimilation Framework

Lars Nerger
Alfred Wegener Institute, Bremerhaven, Germany

Special thanks to:
Himansu Pradhan, Martin Losch
Overview

• Overview of ensemble data assimilation

• Data assimilation software PDAF (Parallel Data Assimilation Framework)

• Implementation example MITgcm
Overview of Ensemble Data Assimilation
Data Assimilation – Motivation

Data Assimilation

Combine both information source to obtain better estimate of system state

**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
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

Unstructured mesh in North-east Atlantic
Observations

Measure different fields in the Ocean

- Remote sensing
  - E.g. surface temperature, salinity, sea surface height, ocean color, sea ice concentrations & thickness

- In situ
  - 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)

Available T and S profiles during July 2008

Scanfish and CTD profiles

MARNET stations
**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!
Leads to different filter methods
Ensemble Covariance Matrix

- Ensemble represents state estimate and its uncertainty
- Uncertainty information (variances + covariances)
- Generated dynamically by propagating ensemble of model states

![Map of std. deviation for total log Chlorophyll]
Data Assimilation Software

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
- ~280 registered users; community contributions

Open source:
Code and documentation available at

http://pdaf.awi.de

Offline coupling – separate programs

**Model**

- **Start**
- **Initialize Model**
  - generate mesh
  - Initialize fields
- **Do i=1, nsteps**
- **Time stepper**
  - consider BC
  - Consider forcing
- **Post-processing**
- **Stop**

**Assimilation program**

- **Start**
- **read ensemble files**
- **analysis step**
- **write model restart files**
- **Stop**

For each ensemble state:
- Initialize from restart files
- Integrate
- Write restart files
- Read restart files (ensemble)
- Compute analysis step
- Write new restart files
Online Coupling

Single program

Model
- initialization
- time integration
- post processing

Observations
- obs. vector
- obs. operator
- obs. error

Generic PDAF Core

Ensemble Filter
- initialization
- analysis step
- ensemble transformation

- - - - - Explicit interface
- - - - - Indirect exchange (module/common)

Lars Nerger

Tutorial: Ensemble Data Assimilation with PDAF
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

Lars Nerger
Tutorial: Ensemble Data Assimilation with PDAF
Implementing Ensemble DA

Example of MITgcm
MITgcm extension for Data Assimilation

**Additions to program flow**

- **Add ensemble parallelization**
  - Initialize parallel.
  - Initialize Model
    - Initialize coupler
    - Initialize grid & fields
  - Init_PDAF
  - Do $i=1, nsteps$
    - Time stepper
      - in-compartment step coupling
    - Assimilate_PDAF
    - Post-processing
  - Finalize_PDAF

**Changes in MITgcm source code**

- **Changes in** (eeboot_minimal.F)
  - subroutine call added (the_main_loop.F)
  - subroutine call added (the_main_loop.F)
  - subroutine call added (the_main_loop.F)

**For convenience:**

- **All changes included in MITgcm repository version**
- **PDAF interface routines**
  - activated by preprocessor setting
    - -DUSE_PDAF

**Additions to**

- **eeboot_minimal**:
  - also change the index of STDOUT file
- **the_main_loop**:
  - timers for PDAF calls
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
- Observation handling
- Write model fields into PDAF’s state vector and back into model fields

**PDAF release includes set of model binding routines for MITgcm**

- for a simple test case
- just download and adapt for your needs
• 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)
Parallelization of Assimilation Program

We use MPI (Message Passing Interface)

- It’s the standard for highly scaling parallelization
- MITgcm uses MPI (like most large-scale models)

Change of parallelization is fully implemented for MITgcm!
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
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)
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)
```

**Fully implemented for MITgcm!**
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
Tutorial: Ensemble Data Assimilation with PDAF

Lars Nerger

Ensemble Filter Analysis Step

case-specific operations

Model interface

Ensemble of state vectors

X

Filter analysis
update ensemble assimilating observations

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

Vector of observations

y

Observation operator

H(...)

Observation error covariance matrix

R

Observation module

For localization:
Local ensemble
Local observations
Ensemble Filter Analysis Step

- **Case-specific call-back routines**
- **Model interface**
  - Ensemble of state vectors \( X \)
- **Filter analysis**
  - Update ensemble assimilating observations
- **Observation module**
  - \( g2l\text{\_state}() \)
  - Local ensemble
  - Local observations
  - \( \text{init\_obs\_l}() \)

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

- **Vector of observations** \( y \)
- **Observation operator** \( H(...) \)
- **Observation error covariance matrix** \( R \)
- **Init obs**
- **Obs op**
- **Prod obs R**
Summary

Ensemble Data Assimilation with PDAF

- augment program for ensemble data assimilation
- assimilation methods provided by PDAF
- model-binding routines required
  - provided for MITgcm for test case
  - easy to code yourself
- PDAF is available as free open-source

Thank you!

Lars.Nerger@awi.de  http://pdafl.awi.de